AŇO 2022

3300 - 003381 / 20 - 003

Creado: 27-09-2022

Iniciador: MESA DE ENTRADAS - INFORMATICA
FERELLA NICOLAS - PIZZIO PABLO

Extracto: E/ TRABAJO FINAL DE TESINA DE GRADO DENOMINADA “IDENTIFICACION DE GENES ASOCIADOS A VIRULENCIA DE S. PYOGENES UTILIZANDO TECNICAS DE MACHINE LEARNING SOBRE SECUENCIAS DE GENOMA COMPLETO” POR “IDENTIFICACION DE PROPIEDADES BIOLOGICAS EN ORGANISMOS UTILIZANDO TECNICAS DE MACHINE LEARNING SOBRE SECUENCIAS DE GENOMA COMPLETO”, DIRIGIDA POR LA PROFESORA CLAUDIA PONS, CON LA ASESORIA PROFESIONAL DE LA DOCTORA JOSEFINA CAMPOS. ---

La Plata, 26 de Septiembre de 2022

Señor Decano de la Facultad de Informática de la UNLP
Dr. Marcelo Naiouf
S_____/_____D

Nos dirigimos a usted para solicitarle que reciba, en formato digital, el informe final de la tesina de grado titulada “Identificación de propiedades biológicas en organismos utilizando técnicas de Machine Learning sobre secuencias de genoma completo”.

Sin otro particular lo saludamos atentamente.

Autorizo mi notificación por correo electrónico. –

Pablo Román Pizio

Nicolás Ferella

Avalamos la presente solicitud.

Dra. Josefina Campos

Dra. Claudia PONS

UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Identificación de propiedades biológicas en organismos utilizando técnicas de Machine Learning sobre secuencias de genoma completo

AUTORES: Nicolás Ferella – Pablo Román Pizio

DIRECTOR/A: Claudia Pons

ASESOR/A PROFESIONAL: Josefina Campos

CARRERA: Licenciatura en Informática

Resumen

El avance de la tecnología y los procesos de secuenciación de genomas de las últimas décadas ha puesto al alcance de investigadores grandes volúmenes de datos biológicos, los cuales resultan difíciles de analizar debido a su escala. Se realizó un software que facilita el análisis de propiedades biológicas sobre cientos o miles de secuencias de genomas completos de un organismo mediante técnicas de Machine Learning, permitiendo al investigador realizar predicciones y encontrar los genes de mayor impacto que, en caso de no estar clasificados hasta la fecha, resultan de interés para su posterior análisis en laboratorio.

Palabras Clave

ADN, Docker, genética, genoma, Inteligencia Artificial, Machine Learning

Conclusiones

La herramienta logra reducir la brecha entre la biología y la informática, al permitir analizar grandes volúmenes de información mediante una interfaz amigable a biólogos con poca experiencia en Inteligencia Artificial.

Trabajos Realizados

Se encontraron soluciones informáticas a problemas técnicos reales, partiendo del entendimiento del dominio de la biología con ayuda de especialistas en el tema, para poder procesar información cruda de gran tamaño y complejidad. A su vez se extendió la funcionalidad del software para agilizar el trabajo del día a día de los científicos, como son la generación de archivos multifasta, el alineamiento de los mismos y la generación de árboles IQTREE sobre los genes de mayor impacto en un estudio.

Trabajos Futuros

Extender la plataforma para poder indicar una lista de genomas alojados en la nube y así poder utilizar datos públicos que se encuentren en diferentes portales (ej. NCBI). Extender el funcionamiento para distribuir el entrenamiento a nivel hardware, reduciendo los tiempos de ejecución. Extender el comportamiento de las redes neuronales para permitir predicciones sobre preguntas no binarias, dando lugar a la investigación de propiedades biológicas que requieren una clasificación más compleja.

Fecha de la presentación: Septiembre 2022

Identificación de propiedades biológicas en organismos utilizando técnicas de Machine Learning sobre secuencias de genoma completo.

Ferella, Nicolás - Pizio, Pablo Román

18 de septiembre de 2022

Resumen

El avance de la tecnología y los procesos de secuenciación de genomas de las últimas décadas ha logrado poner al alcance de investigadores de todo el mundo grandes volúmenes de datos biológicos, que debido a su gran escala, los mismos resultan difíciles de analizar en su totalidad, por lo cual es intuitivo pensar en Inteligencia Artificial para trabajar con dicha información.

Con el objetivo de disminuir la brecha existente entre el investigador y las herramientas de Inteligencia Artificial, se desarrolló un software que permite crear un espacio de trabajo para un organismo biológico, realizar el procesamiento de los genomas correspondientes y permitir la creación y entrenamiento de modelos de Machine Learning desde una interfaz gráfica.

Los modelos entrenados luego se analizan para buscar qué patrones determinan el resultado de la propiedad biólogica a investigar sobre el organismo biológico en cuestión, y así encontrar los genes de mayor impacto en las predicciones del modelo, permitiendo al investigador el posterior análisis en laboratorio de un gen deseado.

Índice general

Capítulo 1

Introducción

La salud y la enfermedad son parte integral de la vida, del proceso biológico y de las interacciones medio ambientales y sociales. Se entiende a la enfermedad como la pérdida de la salud, cuyo efecto negativo es consecuencia de una alteración estructural o funcional de un órgano a cualquier nivel. Muchas enfermedades son causadas por organismos externos, como virus y bacterias.

Para comprender la epidemiología en términos de cómo se diseminan, las características que poseen, la importancia de ciertos genes y la gravedad de las enfermedades que pueden producir, los investigadores deben realizar análisis de la estructura biológica de los organismos estudiados. Esto se logra por medio de la recolección de muestras en sujetos infectados, para su posterior aislamiento y cultivo en laboratorio, permitiendo luego realizar la secuenciación del genoma de los mismos.

Los avances de la tecnología han traído maneras más eficientes en cuanto a costos y tiempos de secuenciación de genomas; procesos que costaban miles de dólares para realizar sobre una bacteria ahora cuestan menos de cien.

A su vez, los tiempos de secuenciación han disminuido considerablemente y su precisión ha aumentado, permitiendo pasar en las últimas décadas de un reducido numeros de muestras secuenciadas a cientos de miles.

Organizaciones como GenBank, la cual es una base de datos de acceso público de secuencias de nucleótidos de más de 100.000 organismos diferentes, han puesto al alcance de investigadores alrededor del mundo grandes y variados volúmenes de datos biológicos que de otro modo les serían imposibles de recolectar, abriendo las puertas a nuevos tipos de estudios.

Es dentro de este marco que se gesta la tesina: la realización de un software para investigadores que ayude en el análisis de propiedades biológicas

sobre cientos o miles de secuencias de genomas completos de un organismo mediante técnicas de Machine Learning, permitiendo realizar predicciones y encontrar los genes de mayor impacto, que en caso de ser genes no clasificados hasta la fecha, resultan de interés para su posterior análisis en laboratorio.

El desarrollo de este trabajo fue en conjunto con investigadores del Instituto Nacional de Enfermedades Infecciosas (INEI) y el Centro Nacional de Genómica y Bioinformática (CNGB) de la Administración Nacional de Laboratorios e Institutos de Salud "Dr. Carlos Malbrán" (ANLIS - Malbrán), realizando una primera investigación y caso de uso de la herramienta sobre la invasividad de los genes de la bacteria *S. pyogenes*, contenidos en 1638 genomas.

Esta tesis está organizada de la siguiente forma: en el capítulo 2 se describen conceptos de genética y biología, en el capítulo 3 se introducen conceptos de Machine Learning y *redes neuronales artificiales* (RNA), en el capítulo 4 se describe el estado del arte en relación a los diversos desarrollos y trabajos que han abordado el tema que es objeto de estudio, el capítulo 5 detalla el análisis y caso de uso que gestó la herramienta descripta en el capítulo 6; el capítulo 7 muestra el flujo y uso general de la herramienta realizada. Finalmente, en el capítulo 8 se presentan las conclusiones y líneas de trabajo futuro.

Capítulo 2

Biología

En este capítulo se introducen conceptos de biología necesarios para comprender el funcionamiento y la composición de las bacterias, conceptos que en capítulos subsiguientes nos permitirán desarrollar el objetivo de este trabajo: mostrar como estas forma de vida pueden ser analizadas bajo técnicas actuales de inteligencia artificial.

2.1. Organismos: definición biológica

En biología, un organismo es cualquier entidad individual que contiene toda propiedad necesaria para considerarse una forma de vida. Todo organismo posee la capacidad de reproducirse, crecer y desarrollarse, mantenerse, y además poseen algun grado de respuesta a un estímulo externo. Los organismos pueden dividirse en 3 reinos: El reino de las eucariotas (organismos cuyas celulas poseen un núcleo), el reino de las arqueas y el reino de las bacterias.

2.1.1. Bacteria

Las bacterias son un tipo de célula biológica. De un tamaño de apenas unos micrones de largo, las bacterias están dentro de las primeras formas de vida de la Tierra. Están presentes dentro de la mayoría de los hábitats que se encuentran en la Tierra.

Las bacterias puede ser tanto beneficiosas como perjudiciales para la salud humana. Las “amistosas” comparten el espacio y los recursos dentro de nuestro organismo y tienden a ser de ayuda. En comparación, hay cerca de 10 veces más células microbianas que humanas en nuestro cuerpo.

El intestino humano es un hábitat especial para las bacterias, ya que se encuentra una plena cantidad de nutrientes para mantenerlos. Las bacterias intestinales, como las cepas útiles de *E. coli* y *Streptococcus*, ayudan en la digestión, evitan la colonización por patógenos dañinos y ayudan a desarrollar el sistema inmunológico.

Varias bacterias, que van desde el Streptococcus del grupo A, *Clostridium perfringens*, *E. coli* y *S. aureus*, pueden causar una infección de tejidos blandos grave llamada fascitis necrotizante (a veces llamada bacteria carnívora), la cual afecta los tejidos que rodean los músculos, los nervios, la grasa y los vasos sanguíneos; es tratable, siempre y cuando se detecte temprano.

2.2. ADN, ARN, genoma y gen

El ácido desoxirribonucleico (ADN) es una molécula compuesta por dos cadenas de polinucleótidos que se entrelazan entre sí, formando una doble hélice que contienen las instrucciones genéticas para el desarrollo, funcionamiento, crecimiento y reproducción de todos los organismos vivos conocidos, y como también de numerosos virus. El ácido ribonucleico (ARN), es una molécula polimérica, la cual es esencial en varios roles biológicos para la codificación, decodificación, regulación y expresión de genes. Tanto el ADN como el ARN son conocidos como ácidos nucleicos los cuales, junto a las proteínas, los lípidos y los carbohidratos complejos, conforman los 4 tipos principales de macromoléculas esenciales para todo tipo de vida.

Cada molécula de ADN está compuesta por miles de copias de 4 bases específicas ricas en nitrógeno:

- Adenina (A)
- Guanina (G)
- Citosina (C)
- Timina (T)

De la misma manera que una secuencia de caracteres puede proveer información en forma de palabras o sentencias, la secuencia de estas bases es la que contiene el mensaje del ADN, el cual provee la información correspondiente para producir las proteínas necesarias. La secuencia total de ADN de un organismo se denomina genoma. Un gen es una secuencia de ADN dentro del genoma que codifica un producto, el cual puede ser proteína (como es el caso de la mayoría de los genes) o puede ser ARN.

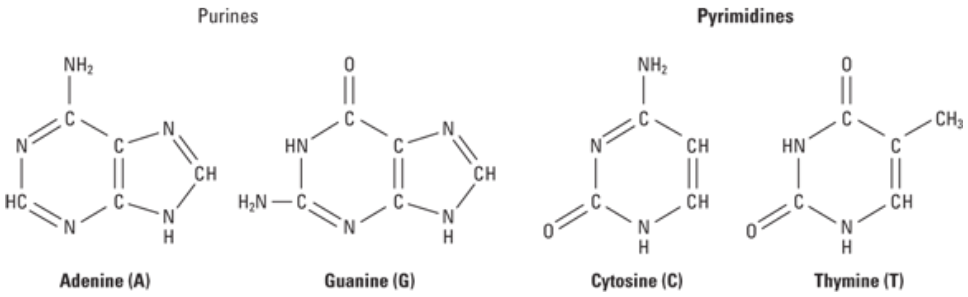


Figura 2.1: Las cuatro bases del ADN

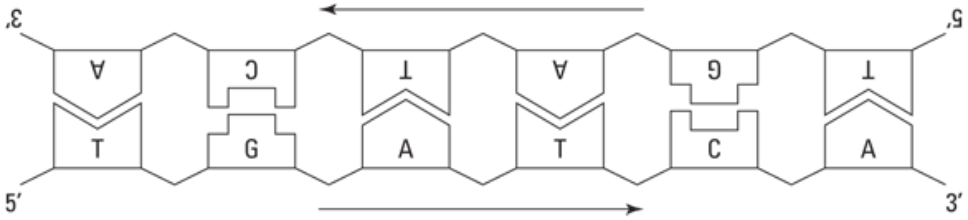


Figura 2.2: representación de enlaces de nucleótidos. La tianina representada con T, Adenina con A, guanina con G, citosina con C

Estas bases no pueden enlazarse entre ellas por si solas, para ello, deben adherirse a una molécula de fosfato y una de deoxiribosa, creando así el bloque básico del ADN, el nucleótido. Dentro de cada uno de estos nucleótidos, se encuentran 2 de las 4 bases enlazadas. Este enlace no es aleatorio, cada base puede solo enlazarse con otra base específica. La timina se enlazará con la adenina, la guanina se enlazará con la citosina, y viceversa. Esto se conoce como bases complementarias.

2.3. Cromosomas

Los cromosomas (del griego chroma, color y soma, cuerpo o elemento) es una estructura organizada formada por cadenas de ADN y proteínas, las cuales contienen la mayor parte de la información genética de un organismo. Los cromosomas fueron observados por primera vez en células de plantas por el botánico Karl Wilhelm Von Nägeli en el el siglo XIX. Hoy en día sabemos que todo organismo tiene cromosomas y que su ubicación y cantidad varía dependiendo la especie. Las bacterias, en su mayoría poseen un solo cromosoma en forma de anillo dentro de la célula, a diferencia de los humanos, los cuales poseen un total de 46 cromosomas en el núcleo de cada célula. El

número de cromosomas es generalmente consistente dentro de una misma especie, pero puede variar a veces entre tipos de organismo, desde 1 hasta más de 200. Los cromosomas permiten encapsular el ADN de un organismo de manera eficiente para la posterior distribución de genes durante la mitosis y meiosis.

2.4. Genética

Habiendo ya introducido el concepto de organismos biológicos y ADN, podemos comenzar a adentrarnos en el concepto de genética.

La genética es la rama de la biología que se ocupa del estudio de los genes, la variación genética y la herencia en los organismos. Un gen es la unidad básica en la herencia de los organismos, compuesta por una secuencia de nucleótidos dentro del ADN o ARN, y codifica la síntesis de un producto génico, ya sea un ARN o una proteína. Para la síntesis, una célula usa una porción de su ADN como una plantilla para crear un ARN. Dicha secuencia de nucleótidos se usa para generar una secuencia de aminoácidos en una proteína; esta traducción entre secuencias de nucleótidos y secuencias de aminoácidos se conoce como código genético.

2.5. Representación Computacional del ADN

Ya comprendiendo las diferentes partes que componen el ADN, se puede observar que el mismo puede interpretarse como una cadena de bases, las cuales pueden representarse con 4 letras, una para cada posible base (A=Adenina, G=Guanina, C=Citosina, T=Timina), permitiendo que el mismo pueda ser descripto como una cadena de caracteres. Uno de los métodos más utilizados actualmente para realizar las secuenciaciones es por medio de la plataforma Illumina, la cual se caracteriza básicamente por la ejecución de los siguientes procesos:

- La amplificación de los fragmentos de ADN para la generación del clúster (colonias del mismo fragmento) se realiza mediante el método de PCR en puente.
- La detección de bases en la secuenciación se hace a través de etiquetas fluorescentes.

La generación del clúster se logra mediante el método de amplificación en puente. Los fragmentos de ADN se colocan sobre una superficie sólida de

vidrio separada por carriles. Cada carril está completamente recubierto por oligonucleótidos complementarios a los adaptadores de cada fragmento que se va a secuenciar, por lo que permiten que cada fragmento se pueda anclar a la celda de flujo.

Al finalizar la amplificación clonal, se retiran todas las hebras reversas y quedan únicamente las hebras idénticas a las originales. En este punto, en la placa se introducen nucleótidos modificados con etiquetas fluorescentes específicas para cada tipo. Los nucleótidos utilizados presentan una modificación química (terminadores reversibles) que evita la unión de más de un nucleótido marcado en cada sitio de reacción. Cada vez que una base se adhiere emite una fluorescencia propia que permite su identificación. Este paso se repite simultáneamente con todas las hebras del mismo clúster de forma paralela hasta completar la secuenciación.[1]. El resultado de la secuenciación será un archivo de tipo FASTQ.

FASTQ es un formato de texto para almacenar tanto una secuencia biológica como sus puntuaciones de calidad correspondientes. Tanto la letra de secuencia como la puntuación de calidad están codificadas con un solo caracter ASCII. El mismo está compuesto por 4 secciones, o líneas.

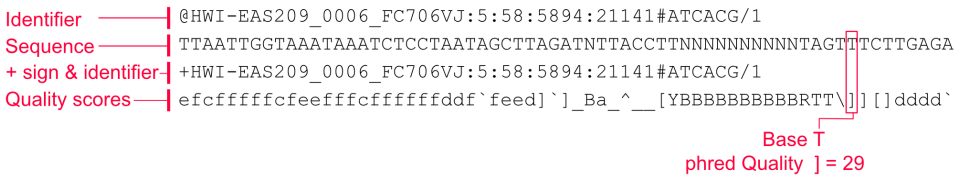


Figura 2.3: Ejemplo de formato FASTQ

La línea 1 comienza con el caracter "@" y va seguida de un identificador de secuencia y una descripción opcional. Esta línea normalmente contiene información específica de la tecnología utilizada para la secuenciación. La línea 2 es la secuencia de letras correspondientes a la secuencia de bases. La línea 3 comienza con un caracter "+", marcando el final de la secuencia de ADN y, opcionalmente, puede contener nuevamente la misma información que la línea 1. La línea 4 codifica los valores de calidad de la secuencia en la línea 2 y debe contener la misma cantidad de símbolos que letras en la secuencia. Cada letra corresponde a una puntuación de calidad. Aunque puede haber diferentes definiciones de los puntajes de calidad, un estándar de facto en el campo es usar "puntajes de calidad Phred". Estos puntajes

representan la probabilidad de que la base sea incorrecta. Formalmente,

$$Q_{fred} = -10log_{10}e$$

dónde e es la probabilidad de que la base se haya secuenciado incorrectamente. Dado que la puntuación está en una escala logarítmica negativa, cuanto más alta sea la puntuación, más improbable será que la base se califique como incorrecta.

Capítulo 3

Redes Neuronales Artificiales

En este capítulo se introducen conceptos sobre *redes neuronales artificiales* (RNA), una herramienta fundamental en el desarrollo de este trabajo.

Las RNA constituyen uno de los tipos de modelos posibles dentro de *Machine Learning*, una rama de la *inteligencia artificial* dedicada al estudio de técnicas que permitan que algoritmos computacionales aprendan mediante la experiencia.[2, 3]

3.1. Definición

Las RNA son un modelo computacional inspiradas en el funcionamiento de las redes neuronales del cerebro humano. Su propósito es generalizar comportamientos a partir de un conjunto de datos de entrenamiento, y así obtener un modelo que permita hacer predicciones para un conjunto de datos más amplio.

Al igual que una red neuronal biológica, las RNA están formadas por un conjunto de neuronas, las cuales se conectan entre sí y se agrupan por capas. Las neuronas de cada capa reciben información de las neuronas de la capa anterior y envian información a las neuronas de la capa siguiente, formando una estructura de grafo dirigido.

3.2. Componentes

3.2.1. Capas

Las RNA se componen por un conjunto de capas, donde cada capa posee una o más neuronas. Se pueden distinguir tres tipos de capas: capa de

entrada, capa de salida y capa oculta.

- La **capa de entrada** representa los datos de entrada real de la red neuronal y en la cual no se produce procesamiento.
- Las **capas ocultas** son las capas que reciben información de las capas anteriores, la procesan y envían nueva información a las capas posteriores. Una RNA puede tener cero o n capas ocultas.
- La **capa de salida** es la última capa, y su valor de salida es el resultado final de la RNA.

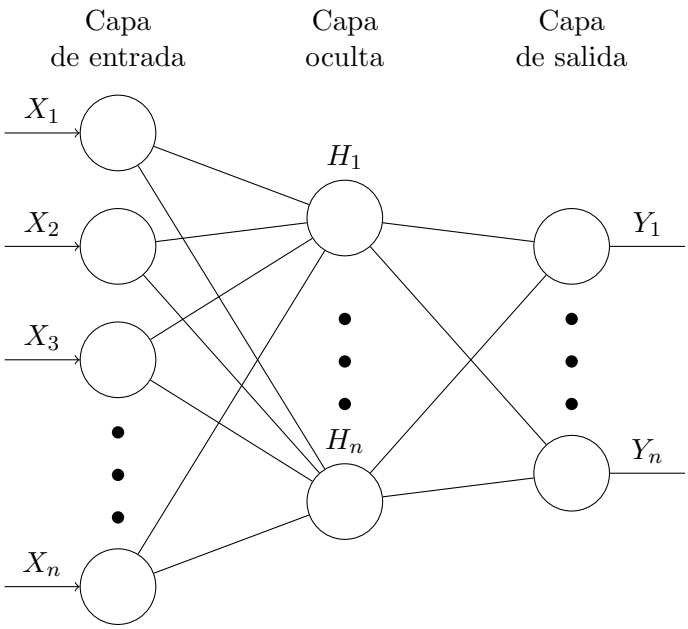


Figura 3.1: Ejemplo de arquitectura de una RNA

En el caso de definir una RNA con una sola capa, la capa de entrada coincidiría con la capa de salida y no habrían capas ocultas.

La cantidad de capas y neuronas por capa que contenga una RNA se determina según la complejidad del problema a resolver, muchas veces encontrando a prueba y error la arquitectura necesaria para cumplir el objetivo. Cuando se utiliza una gran cantidad de capas ocultas para armar una RNA, se puede clasificar al algoritmo dentro de *deep learning*, una subclase de *machine learning*.

3.2.2. Neuronas y conexiones

Una neurona artificial es la unidad básica de procesamiento dentro de una RNA. Al igual que una neurona biológica recibe una serie de impulsos nerviosos, una neurona artificial recibe valores de entrada con los cuales realiza un cálculo y genera un valor de salida. Una neurona artificial es en realidad una función matemática.

Los valores de entrada de una neurona que no pertenece a la capa de entrada, son los valores de salida de las neuronas de la capa anterior. El valor de salida de una neurona que no pertenece a la capa de salida es uno de los valores de entrada de las neuronas de la capa posterior. De esta manera se forman *conexiones* entre las neuronas de las capas adyacentes.

Cada conexión tiene asociado un *peso*, que determina la importancia de un valor de entrada para una neurona. El cálculo que realiza una neurona es una *suma ponderada* de los valores de entrada, ponderando cada valor con el peso de la conexión.

A esta suma ponderada se le suma un valor más denominado *bias*, el cual es otro parámetro de entrada de la neurona. Con este parámetro adicional se forma dentro de la neurona una *función lineal*. La suma ponderada determina la pendiente de la línea recta y el bias el punto de corte con el eje y.

Para completar el cálculo que realiza la neurona, es necesario procesar el resultado de la función lineal por una *función de activación*, la cual es una función no lineal que permite generar resultados no lineales. La función de activación se elige según el problema a resolver y algunas de las más comunes son las funciones *RELU*, *SIGMOIDE* o *TANH*. El resultado de la función de activación es el valor de salida de la neurona.

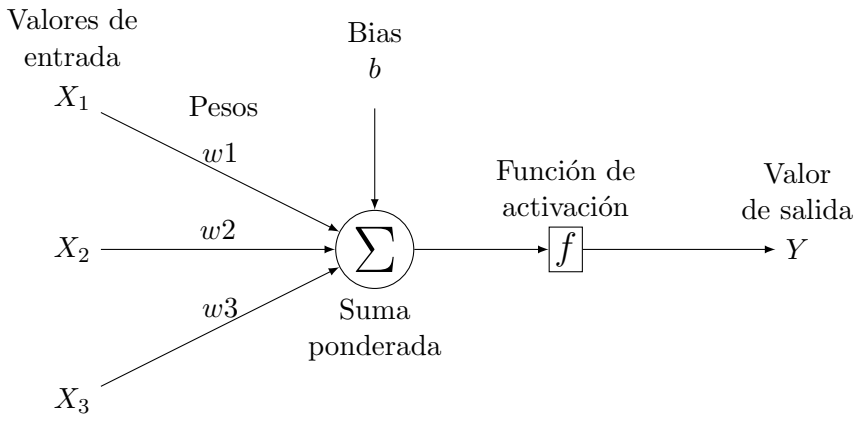


Figura 3.2: Estructura de una neurona

3.3. Aprendizaje

Una RNA debe tener los pesos de las conexiones con ciertos valores para que resuelva el problema para el cual es modelada. La parte interesante de estos algoritmos es que encuentren por si solos cuáles son los valores adecuados para estos parámetros, este proceso es lo que se denomina *aprendizaje*.

3.3.1. Tipos de aprendizaje

Existen dos tipos de aprendizaje principales:

- Supervisado: esta técnica utiliza un conjunto de datos de entrenamiento que consta de un vector de datos de entrada y un vector con el resultado esperado para cada dato de entrada. La idea general de estos algoritmos es iterar por cada dato de entrada y comparar el resultado obtenido con el resultado esperado para ajustar los pesos de las conexiones y en las próximas iteraciones obtener una mayor aproximación a los resultados esperados. De esta manera el modelo aprende y cuando termine la etapa de entrenamiento podrá ser usado para resolver el problema para el cual fue modelado.

- No supervisado: con esta técnica el modelo aprende mediante el reconocimiento de patrones sobre un conjunto de datos de entrenamiento para los cuales no se dispone ningún resultado esperado. En resumen, el objetivo de estos algoritmos es estudiar la estructura de los datos y así utilizarlos, por ejemplo, para la segmentación de los datos en

distintos conjuntos, el reconocimiento de irregularidades o también la simplificación de los datos.

3.3.2. Proceso de aprendizaje

La técnica de aprendizaje utilizada en este trabajo es el aprendizaje supervisado. Este proceso consiste en encontrar cuáles son los valores que deben tener los pesos de las conexiones de la red para que la misma tenga el comportamiento deseado. Para lograr este objetivo se realizan los siguientes pasos:

1. Definir estado inicial de la RNA inicializando los pesos de las conexiones con valores aleatorios.
2. Realizar una iteración alimentando al modelo con valores de entrada.
3. Comparar el resultado obtenido con el resultado esperado para calcular el error.
4. Ajustar los pesos correspondientes para corregir el error.
5. Volver al paso 2 y repetir hasta iterar sobre todos los datos de entrenamiento.

3.3.3. Conjunto de datos

Para realizar la técnica de aprendizaje supervisado es necesario disponer de un conjunto de datos donde cada dato esté etiquetado con el resultado esperado. Por ejemplo: si el objetivo es entrenar una RNA que determine si un número es par o impar, el conjunto de datos sería: un conjunto de números donde cada uno tiene una etiqueta que especifica si es par o impar.

El objetivo es que la RNA encuentre patrones para cada posible etiqueta del conjunto de datos, esto requiere que el conjunto sea grande, ya que si son pocos los datos sobre los cuales el modelo realiza el entrenamiento, se dificulta la tarea de identificar patrones. Que un conjunto tenga pocos datos o la cantidad necesaria depende de la complejidad del problema a resolver.

También es indispensable que el conjunto de datos esté balanceado, es decir, que haya una cantidad similar de datos para cada etiqueta posible. Si el balanceo no se cumple, la RNA no identificará correctamente los patrones que clasifican a un dato a una u otra etiqueta, y probablemente cualquier dato que se quiera evaluar luego del entrenamiento para obtener un resultado será clasificado con la etiqueta para la cual se dispone la mayor cantidad de

datos de entrenamiento. Continuando con el ejemplo anterior: si el conjunto de números incluye 90 números pares y 10 impares, la RNA no identificará correctamente cuál es el patrón que determina que un número sea par o impar, sesgando futuras evaluaciones de cualquier número a que el resultado determinado sea probablemente el de la etiqueta con mayor cantidad de datos asociados.

A la hora de realizar el entrenamiento es conveniente dividir el conjunto de datos en dos partes: una parte que se use para el entrenamiento en si y otra parte para probar el entrenamiento y aprendizaje realizado. De esta manera se puede calcular la tasa de acierto de la RNA, dado que al realizar la prueba se usan datos que están etiquetados con el resultado esperado pero que la RNA no conoce, entonces se compara el resultado obtenido con el esperado y se obtiene la tasa de acierto. Una posible división del conjunto de datos es usar un 85/90 % para entrenamiento y el 15/10 % restante para probar.

3.4. Técnicas de Machine Learning

3.4.1. Feature engineering

Se conoce como Feature engineering, o ingeniería de datos, al proceso de seleccionar, manipular y transformar los datos de entrada sin procesar que se utilizan para crear un modelo predictivo mediante el aprendizaje automático o el modelado estadístico. El objetivo es mejorar el rendimiento de los algoritmos de aprendizaje. A continuación, se hará una breve descripción de diferentes técnicas contenidas dentro de este principio.

Selección de características

En Machine Learning, la selección de características es el proceso de seleccionar un subconjunto de características (variables, predictores) para su uso en la construcción de modelos de Machine Learning.

Esta técnica es utilizada principalmente por las siguientes razones:

- Simplificación de modelos con el fin de hacerlos más sencillos de interpretar para los usuarios/investigadores.
- Mejorar los tiempos de entrenamiento.
- Evitar la maldición de la dimensionalidad (curse of dimensionality)[1]

[1]La maldicion de la dimensionalidad ocurre en dominios como el análisis numérico,

- Mejorar la compatibilidad de los datos con la clase del modelo de aprendizaje a utilizar.

La premisa central del uso de esta técnica es que los datos sin procesar contienen características redundantes o irrelevantes, las cuales pueden ser removidas sin incurrir en pérdida de información importante. Cabe aclarar que la redundancia y la irrelevancia son dos nociones distintas, ya que una característica relevante puede ser redundante en la presencia de otra característica relevante con la que está fuertemente correlacionada. Los casos típicos para la aplicación de la selección de características incluyen el análisis de textos escritos y matrices de ADN, donde hay miles de características y solo decenas o cientos de muestras.

La selección de características debe distinguirse de la *extracción de características*, ya que esta crea nuevos conjuntos de datos a partir de los datos originales, mientras que la selección de características devuelve un subconjunto de los mismos.

Extracción de características

La extracción de características es un proceso de reducción de la dimensionalidad mediante el cual un conjunto inicial de datos sin procesar se reduce a uno o más grupos, al combinar variables en características. El análisis de datos con una gran cantidad de variables generalmente requiere una gran cantidad de memoria y poder de cómputo, además de poder causar que un algoritmo de clasificación se ajuste en exceso a las muestras de entrenamiento, generalizando mal luego a las muestras nuevas. La extracción de características es un término que engloba los métodos de construcción de combinaciones de variables utilizados para sortear estos problemas, reduciendo efectivamente la cantidad de datos que deben procesarse y, al mismo tiempo, describiendo de manera más precisa y completa el conjunto de datos original.

---

el muestreo, la combinatoria, el aprendizaje automático, la minería de datos y las bases de datos. Específicamente en los problemas de aprendizaje automático que implican el aprendizaje de un "estado de la naturaleza" a partir de un número finito de muestras de datos en un espacio de muchas de características, se requiere una enorme cantidad de datos de entrenamiento para garantizar que haya varias muestras con cada combinación de valores. En un sentido abstracto, a medida que crece la cantidad de características o dimensiones, la cantidad de datos que necesitamos para generalizar con precisión crece exponencialmente.

3.5. Conclusiones

En este capítulo se explicaron conceptos básicos sobre *Machine Learning*, que se necesitan para el entendimiento de este trabajo. En los próximos capítulos se verá qué relación existe con la biología, qué otros trabajos han combinado estas dos ramas, y cómo se utilizan las *redes neuronales artificiales* para resolver el problema que se plantea en este trabajo.

También quedará expuesto como es el proceso de desarrollo de una RNA, desde la búsqueda de la gran cantidad de datos necesarios, la preparación de los mismos para que sean compatibles con una RNA (utilizando programas de bioinformática ya existentes y otros procesos propios que los estructuren de la manera requerida), cómo se logra obtener el diseño de una arquitectura de una RNA realizando pruebas con distintas arquitecturas y parámetros posibles hasta que el entrenamiento concluya en un modelo con una tasa de acierto alta, siempre teniendo en cuenta el balanceo de los datos para las posibles etiquetas y así no sesgar al modelo.

Capítulo 4

Estado del arte

Ya introducidos los conceptos necesarios de genética e inteligencia artificial, se caracteriza en el siguiente capítulo el estado del arte en relación a los diversos desarrollos y trabajos que han abordado el tema que es objeto de estudio.

En el trabajo *Identifying genes associated with invasive disease in S. pneumoniae by applying a machine learning approach to whole genome sequence typing data*[4] publicado por Obolski en 2019, se utilizaron técnicas de inteligencia artificial sobre un total de 9032 genes obtenidos de 378 genomas del patógeno *Streptococcus pneumoniae*. En el trabajo se describe como, gracias al uso del algoritmo Random Forest (Random Forest Algorithm; RFA), se lograron identificar los 100 genes de mayor contribuición marginal en 3 sets de muestras diferentes, cruzando luego los datos y llegando a un conjunto reducido de 43 que fueron luego analizados. Durante el análisis de los mismos, se encontró que muchos de ellos ya eran parte de factores de virulencia conocidos, y que otros cumplian funciones que podían causar daños al ADN. La mayoría de los demás genes identificados producían proteínas hipotéticas sin función conocida en la actualidad. Debido a que estos últimos fueron rankeados como parte de los de mayor grado de importancia, los investigadores presumen que pueden estar asociados a procesos involucrados a enfermedades invasivas. Asi mismo, en este estudio los autores describen las limitaciones del método utilizado al no poder diferenciar en los resultados obtenidos las diferentes variaciones de un mismo gen (alelos), por lo que indican que métodos más "sutiles"son necesarios para discernir que versiones son responsables de que cambio fenotípico.

El artículo *Artificial intelligence in clinical and genomic diagnostics*[5] publicado por Raquel Dias y Ali Torkamani en 2019 realiza un recorrido

sobre las posibles aplicaciones de la inteligencia artificial en diagnósticos clínicos y genéticos. En el mismo, se habla sobre la disponibilidad actual de grandes bancos de datos para el entrenamiento, como son grandes colecciones de imágenes médicas con anotaciones o datasets genómicos de miles de muestras médicas, ambos con volúmenes de información imposibles de analizar en su totalidad por el ser humano. Gracias a los avances en los algoritmos de IA y a la mejora del hardware usado para su ejecución, se ha empezado a ver un aumento en su uso y su productividad. Los autores comentan que los algoritmos de inteligencia articicial se han mostrado muy prometedores en una variedad de tareas de genómica clínica, como pueden ser clasificaciones y predicciones del impacto funcional, concluyendo que las herramientas de IA más generalizadas podrán llegar a convertirse en el estándar en estas áreas, especialmente para las tareas de genómica clínica donde la inferencia a partir de datos complejos es una tarea que se realiza con frecuencia.

Los sistemas de IA sintetizan características simples en conceptos más complejos para derivar conclusiones sobre los datos de salud de una manera análoga a la interpretación humana, aunque los conceptos complejos utilizados por los sistemas de IA no son necesariamente conceptos obvios o reconocibles para los humanos.

En el articulo *Type III secretion system effectors form robust and flexible intracellular virulence networks*[6], se describe el trabajo realizado por un grupo de investigadores quienes, utilizando cien variantes de la bacteria *Citrobacter rodentium*, cada una con una combinación diferente de efectores, investigaron cómo cada una de estas infecta las células y el intestino de ratones. En las observaciones, constataron que las proteínas maliciosas actúan cooperando entre ellas y que estas tienen algunas funciones por duplicado para garantizar el éxito en caso de que alguna parte falle. Los investigadores relatan que hasta ese momento se analizaban los efectores uno por uno y que, en su trabajo, se buscó hacerlo en su conjunto, viendo el efecto en la infección y la respuesta del sistema inmune. Los datos conseguidos in vitro y en ratones fueron utilizados para construir un modelo de aprendizaje automático, usando técnicas de inteligencia artificial para poder estudiar las millones de combinaciones de efectores.

Los investigadores pudieron centrar el foco en las variantes más interesantes, descubriendo que hay pequeños grupos de efectores que son esenciales en cualquier circunstancia. Esto significa que cuando se eliminan o bloquean, las bacterias no infectan, lo que supone una "prometedora" diana para futuros tratamientos.

En la tesina de grado *Cliente para plataforma de búsqueda de biomarcadores con valor pronóstico/predictivo de cáncer*[7] desarrollada por Diego

Ariel Martínez, se diseñó y desarrolló una aplicación cliente extensible, portable, de fácil distribución en el marco de una plataforma, cuyo origen se remite a la tesis 'Metodología analítica e integradora para la generación de biomarcadores de pacientes con cáncer de mama sobre la base de perfiles de expresión génica' de maestría en Explotación de datos y descubrimiento de conocimiento de Matías Butti. En dicho trabajo se presenta una plataforma, denominada Bioplat, para la identificación, validación y optimización de biomarcadores en cáncer. La plataforma Bioplat está compuesta de varios componentes y una de los principales es el cliente desarrollado en este trabajo, en el cual se describe la integración del mismo con el resto de la plataforma, tanto desde el punto de vista funcional y de sus requerimientos, como desde lo técnico. Además, se presentan las soluciones de software propuestas a la estructura de la plataforma que permitan fácilmente, que ésta sea extensible por el equipo de desarrollo Bioplat, por desarrolladores que deseen enriquecer la plataforma con ideas y/o desarrollos propios compartidos luego con la comunidad Bioplat o por usuarios finales. Con el cliente integrado a la plataforma, se logró realizar un producto de fácil distribución e instalación para los investigadores. Se definió un marco de extensión consistente, que permite al cliente, y así a la plataforma, enriquecerse en nuevas herramientas para profundizar el estudio de los biomarcadores.

Habiendo encontrado como factor común en estos trabajos una clara observación por parte de los investigadores de los beneficios que las tecnologías actuales pueden aportar a la investigaciones del campo de la biología, siendo una de las mayores barreras el creciente volumen de información y la necesidad de herramientas sistemáticas para una evaluación de la misma, se ha abordado esta problemática en la presente tesina, la cual busca reducir la brecha entre estas dos ciencias al acercar una herramienta que permita cumplir estas tareas a los biólogos con poca experiencia en IA.

Capítulo 5

Caso de uso

El *Streptococcus Pyogenes* ó estreptococo del grupo A (Group A beta-hemolytic Streptococcus, GAS) es una bacteria causante de numerosas enfermedades, entre ellas, la faringoamigdalitis, la escarlatina y el impétigo. El estudio global de enfermedades realizado en el 2010 por The Lancet en conjunto con la Organizacion Mundial de la Salud (WHO) estimó que hay 140.495.000 casos de impétigo en todo el mundo cada año. Esta carga infecciosa lo coloca entre las 50 enfermedades más comunes en todo el mundo. Las infecciones cutáneas por *S. pyogenes* es una de las causas más importantes de morbilidad en entornos con recursos limitados[8].

El GAS también puede provocar enfermedades invasivas graves, como la fascitis necrotizante y el síndrome de shock tóxico estreptocócico, las cuales presentan elevadas tasas de mortalidad, con un estimado de 517.000 muertes anuales a nivel global[9].

Para comprender completamente su epidemiología en términos de cómo se disemina, las características de la cepa, su importancia para la transmisión y la gravedad de la enfermedad que puede producir, los investigadores deben realizar análisis de la estructura biológica de estas bacterias. Esto se logra por medio de la recolección de muestras de personas infectadas para el posterior aislamiento y cultivo en laboratorio, permitiendo luego realizar una secuenciación de la estructura genómica de la cepa.

Como se mencionó previamente, los avances de la tecnología han traído maneras más eficientes en cuanto a costos y tiempos de secuenciación de genomas, permitiendo pasar en las últimas décadas de un reducido número de muestras secuenciadas, a cientos de miles, las cuales hoy se encuentran al alcance de investigadores alrededor del mundo, que de otro modo les serían imposibles de recolectar.

5.1. Descarga y preparación de datos

Para la creación del dataset se comenzó con la búsqueda, identificación y descarga de genomas en formato FASTQ dentro numerosos estudios publicados en la base internacional de datos de secuencias de nucleótidos, llegando a mas de 1600 descargas con un total de 300Gbs en el transcurso de 3 meses. La búsqueda e identificación se realizó en conjunto con investigadores del Centro Nacional de Genómica y Bioinformática (CNGB).

Los archivos FASTQ pueden contener hasta millones de entradas y pueden tener desde varios megabytes, hasta gigabytes de tamaño, haciéndolos demasiado pesados para ser trabajados en grandes sets, por lo que se consideran archivos de salida intermedios, los cuales se utilizan como entrada para herramientas que realizan análisis posteriores. Estos archivos están compuestos por múltiples lecturas de células cultivadas de la muestra biológica a analizar, por lo que el mismo contendrá "trozos" de estas lecturas. Esto permite tener redundancia en las lecturas, la cual es una manera de asegurar o puntuar la calidad de los datos obtenidos, al costo de sumar complejidad a la hora de analizarlo como un tipo de dato homogéneo y continuo. Otro desafío que presentan este tipo de archivos es que su estructura variará dependiendo del sistema y los parámetros que han sido utilizados para realizar la secuenciación de las muestras. Esto nos llevó a buscar realizar un proceso de estandarización de los archivos, para lo cual utilizamos diferentes programas con la supervisión de los científicos del CNGB, para convertir los archivos a formatos con menor redundancia de datos y mayor contenido de metadata.

El primero de estos programas es el *St. Petersburg genome assembler*, o SPAdes. SPades toma de entrada un archivo de tipo FASTQ y, utilizando internamente un grafo dirigido o Grafo de Brujin, permite representar los solapamientos de las múltiples lecturas en una lectura continua. Esta información se devuelve en un nuevo archivo de tipo FASTA.

El archivo FASTA luego es procesado por medio del software Prokka, el cual identifica los genes que la muestra contiene y la proteína que produce, así como también si estas poseen una funcionalidad conocida, o si las mismas son proteínas hipotéticas, y vuelca esta información en un archivo de formato GBF (GenBank flatfile format).

Este archivo está conformado por una sucesión de entradas[1] como las mostradas en la figura 5.2 por cada gen que se haya identificado dentro

[1]La descripción completa de los campos que conforman un archivo GBF puede ser consultada en https://www.ncbi.nlm.nih.gov/genbank/samplerecord/#ProteinIDB

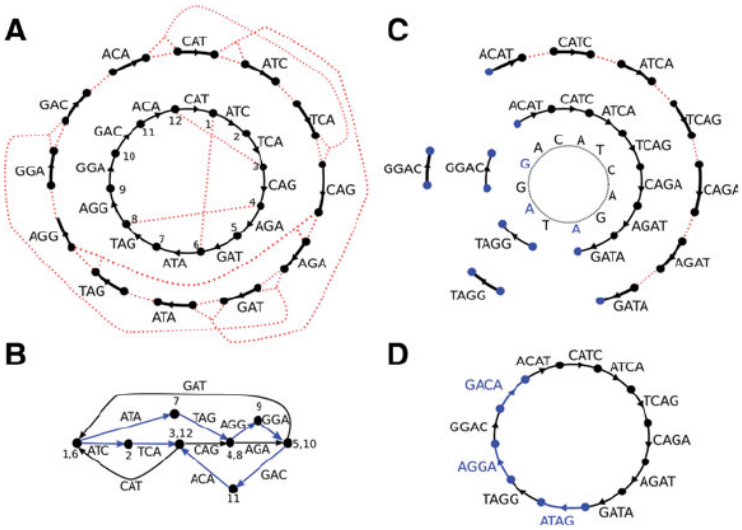


Figura 5.1: *Ejemplo de grafo de Bujin en la alineación de múltiples lecturas. (A) El círculo exterior muestra las diferentes lecturas, y las líneas rojas indican los posibles vértices que se unirán. (B) Grafo resultante de las posibles uniones. (C) Unión de los segmentos hasta llegar al (D) genoma resultante*

de la muestra estudiada, de las cuales las principales utilizadas en nuestro estudio fueron las siguientes:

- CDS: indica el comienzo de los metadatos de un gen, seguido por dos números. Los mismos nos indican el comienzo y final de la cadena de ácidos nucléicos que lo componen dentro del archivo FASTA procesado. En caso de encontrarse en la cadena complementaria, se indica en este campo agregando los números entre paréntesis luego del tag "complement".
- CDS/gene: Indicará el nombre corto del gen en caso de ya existir en la base de datos utilizada de referencia (Genbank en nuestro estudio).
- locus_tag: identificador único para el gen dentro del archivo GBF.
- translation: la traducción a aminoácidos correspondiente a la secuencia de nucleótidos del gen.

```
gene            6244..6885
                /gene="nudL"
                /locus_tag="GFKLDIDM_00007"
CDS             6244..6885
                /gene="nudL"
                /locus_tag="GFKLDIDM_00007"
                /EC_number="3.6.1.-"
                /inference="ab initio prediction:Prodigal:002006"
                /inference="protein motif:HAMAP:MF_01592"
                /codon_start=1
                /transl_table=11
                /product="putative Nudix hydrolase NudL"
                /translation="MKDLLKQYQAKPLGEEKRYAVFLPLILVNDDWHVLYEVRSQHIS
                QPGEVSFPGGQVENQETLQEAAIRETVEELTVDASQIQLWGEIDYLVQSSRTIHCFVG
                QLVVDDWKSIQPNEEVDKVFIVPLRQLLFTDPVYYRLEVTPIETTDFPFDRIRNGKYY
                QFSQEYRSIPFYENLEETIWGMTAQFTKCLTDILNDESIAKKNPSDIEKSKGR"
```

Figura 5.2: *Ejemplo de contenido de un archivo GBF generado por el software Prokka a partir de un archivo FASTA. En el mismo se pueden observar los metadatos que se agregan a cada gen identificado, como el nombre de la proteína que produce si es conocida, y su traducción de cadena de nucleótidos a una cadena proteica*

- product: nombre de la proteína producida por la cadena de nucleótidos del gen. Este tag contendrá el valor "hypothetical protein" en caso de no existir en la base de datos de GenBank.

Se seleccionaron 1638 muestras de *S. pyogenes*, 819 consideradas invasivas debido a los síntomas generados en cada uno de los sujetos de los cuales se extrajo la muestra, y 819 no invasivas. Estas muestras fueron procesadas con los programas descriptos anteriormente para obtener su correspondiente archivo GBF.

5.1.1. Parseo de datos y creación de DB

Ya teniendo un archivo GBF de cada muestra se procedió a analizar la información de los genes de cada una de ellas.

```
1  if 'CDS' in Lines[count]:
2      gene = ''
3      product = ''
4      locus_tag = ''
5      translation = ''
6      gene_parsed = False
7      count += 1
```

```
    while count < len(Lines) and not gene_parsed:
        stripped = ''
        if '=' in Lines[count]:
            stripped =
            ↪  Lines[count].strip().strip('\n').split("=")[1].strip('"')
                gene = stripped if "/gene=" in Lines[count] else gene
        product = stripped if "/product=" in Lines[count] else product
        locus_tag = stripped if "/locus_tag=" in Lines[count] else
        ↪  locus_tag
        if "/translation=" in Lines[count]:
            translation = stripped
            count+= 1
            while (('"' not in Lines[count]) and ('ORIGIN' not in
            ↪  Lines[count]) and ('gene' not in Lines[count])):
                translation+=Lines[count].strip().strip('\n')
                count+= 1
            if (('ORIGIN' not in Lines[count]) and ('gene' not in
            ↪  Lines[count])):
                translation+=Lines[count].strip().strip('\n').strip('"')
            gene_parsed = True
        count+= 1
    gen = {
        "gene":gene,
        "product":product,
        "locus_tag":locus_tag,
        "translation":translation
    }
    genes.append(gen)
```

Extracto 1: *código utilizado en Phyton para el recorrido e identificación de las características de cada gen*

La cantidad de genes encontrados en cada uno de los genomas analizados fue entre 2500 y 3800. Una vez obtenidos los genes de cada muestra, se procedió a realizar un pangenoma[2] de las muestras del estudio. Se obtuvieron más de 158.000 genes diferentes.

[2]El pangenoma (o supragenoma) en biología molecular describe la colección de todos los genes en una especie (aplicado típicamente a bacterias y arqueas, que pueden presentar una gran variación de contenido genético entre cepas estrechamente relacionadas). Es un superconjunto de todos los genes de todas las cepas de una especie

5.1.2. Red neuronal: Creación y entrenamiento

Tecnologías

Para la creación de la red neuronal, se utilizaron diferententes tecnologías, de las cuales realizaremos a continuacón una breve descripción.

- Tensorflow: Biblioteca de software gratuita y de código abierto utilizada para el aprendizaje automático e inteligencia artificial. Se puede usar en una variedad de tareas, pero tiene un enfoque particular en el entrenamiento e inferencia de redes *deep learning*. TensorFlow fue desarrollado por el equipo de Google Brain para uso interno de Google en sus investigaciones Tensorflow puede ser usada en una variedad de lenguajes de programación, siendo los más utilizado Python, Javascript, C++ y Java, lo cual le otorga una gran flexibilidad. Nuestro prototipo fue realizado utilizando Python. Tensorflow utiliza Keras para permitir a los usuarios crear sus propios modelos de *machine learning*.

- Keras: Libreria de software de código abierto que proporciona una interfaz en Python para redes neuronales artificiales. Keras actúa como una interfaz para la biblioteca TensorFlow. Diseñado para permitir la experimentación rápida con redes de deep learning, se enfoca en ser fácil de usar, modular y extensible. Fue desarrollado como parte del esfuerzo de investigación del proyecto ONEIROS *(del inglés Open-ended Neuro-Electronic Intelligent Robot Operating System)* Keras contiene numerosas implementaciones de bloques de construcción de redes neuronales de uso común, como *layers, objectives, activation functions, optimizers* y una gran cantidad de herramientas para facilitar el trabajo con datos de imágenes y texto simplificando la codificación de redes neuronales *deep learning*. Además de las redes neuronales estándar, Keras admite redes neuronales convolucionales y recurrentes.

- Numpy: Una de las bibliotecas de datos de Python más populares para la cual TensorFlow ofrece integración y compatibilidad con sus estructuras. Numpy NDarrays, el tipo de datos nativo de la biblioteca, se convierten automáticamente en TensorFlow Tensors.

Creación de la red neuronal

Utilizando el pangenoma generado de todas las muestras del estudio, se procedió a crear una matriz de pertenencia de genes para cada una de las muestras del estudio el cual, junto a la propiedad de invasividad de cada

una de ellas, fueron utilizadas en el entrenamiento de una red neuronal. Para ello, se importaron las librerías mencionadas anteriormente en python y se procedió a crear un modelo de inteligencia artificial en tensorflow por medio de la libreria keras.

Se comenzaron las primeras pruebas de entrenamiento utilizando una taza de aprendizaje de 0.005 y una cantidad de 5 a 10 capas secuenciales compuestas por una reducción escalonada de neuronas desde 8072 hasta 1.

Debido a la cantidad de datos de entrada (arreglos de más de 158.579 elementos para cada una de las 1638 muestras), los tiempos de ejecución para el aprendizaje del modelo resultaban muy elevados, haciéndolos poco eficientes tanto en tiempo como en recursos. Para mejorar los tiempos de aprendizaje, se buscó utilizar el concepto de alelo[3] en las muestras de entradas. Para ello, utilizando el concepto de *extracción de características* descripto en el capítulo 3, se procedió a identificar los genes que producían la misma proteína como resultado, generando con estos un nuevo arreglo de pertenencia para cada una de las muestras, reduciendo asi los datos de entradas de cada una de ellas de 158.579 a 82.520.

A su vez, como un segundo enfoque, y para buscar mejorar aun más los tiempos de aprendizaje, se utilizó el concepto de selección de características descripto en el capítulo 3. Para ello, se examinaron los datos de entrada, analizando la distribución de genes dentro de las muestras. Durante este proceso, se detectó que muchos de estos alelos podían encontrarse en igual cantidad de muestras invasivas y no invasivas, haciendo que los mismos no tengan un aporte significativo al aprendizaje. Definiendo un nuevo parámetro de configuración, se utilizó un porcentaje de tolerancia máximo en la diferencia de pertenencia entre muestras invasivas y no invasivas. Utilizando un 1 % de tolerancia (diferencia de 16 muestras en nuestro caso de estudio), se redujo el tamaño de cada dato de entrada de 82.520 a 5.218 genes.

En cuanto a la función de activación para las diferentes capas, se obtuvieron los mismos porcentajes de precisión tanto en el uso de las funciones sigmoid como RelU, siendo esta última más eficiente en terminos de tiempo de ejecución.

[3]Un alelo es una variación de la misma secuencia de nucleótidos que codifica la síntesis de un producto génico en el mismo lugar de una molécula de ADN. La mayoría de los alelos producen poco o ningún cambio en la función del producto génico que codifica.

```
f.compat.v1.disable_eager_execution()
lr = 0.0005           # learning rate
nn = [8192, 4096, 1024, 256, 64, 16, 1] #número de neuronas por capa.
# Creamos el objeto que contendrá a nuestra red neuronal, como
# secuencia de capas.
model = kr.Sequential()

# agregamos capas
l1 = model.add(kr.layers.Dense(nn[1], activation='relu'))
l2 = model.add(kr.layers.Dense(nn[2], activation='relu'))
l3 = model.add(kr.layers.Dense(nn[3], activation='relu'))
l4 = model.add(kr.layers.Dense(nn[4], activation='relu'))
l5 = model.add(kr.layers.Dense(nn[5], activation='relu'))
l6 = model.add(kr.layers.Dense(nn[6], activation='relu'))
l7 = model.add(kr.layers.Dense(nn[7], activation='sigmoid'))
# Compilamos el modelo, definiendo la función de coste y el optimizador.
model.compile(loss='mse', optimizer=kr.optimizers.SGD(learning_rate=lr),
↪  metrics=['acc'])
#Entrenamos
model.fit(np.array(X[:1350]), np.array(Y[:1350]), epochs=1000)

print("Evaluate on test data")
results = model.evaluate(np.array(X[1350:1500]), np.array(Y[1350:1500]))
print("Mixed test loss, test acc:", results)
results = model.evaluate(np.array(X[1500:]), np.array(Y[1500:]))
print("All invasive test loss, test acc:", results)
```

Extracto 2: *Ejemplo de modelo de red neuronal testeado para las muestras de streptococus pyogenes del estudio*

Modelos

A continuación se describen 5 modelos diferentes y los resultados obtenidos en el análisis de las muestras de nuestro caso de uso. La cantidad de muestras utilizadas para todos los modelos fue de 1638 genomas, realizando un total de 500 épocas en cada caso.

	Tipo de datos	Función	Capas	Neuronas por capa	LR
Modelo 1	Genes 1 % tol. dif. (9.131)	ReLU	10	85.520, 32.072, 8.192, 4.096, 1.024, 512, 256, 64, 32, 16	0.005
Modelo 2	Alelos 1 % tol. dif. (5.218)	ReLU	8	16.384, 12.268, 8.192, 4.096, 1.024, 256, 64, 16	0.005
Modelo 3	Genes 2 % tol. dif. (2.246)	ReLU	8	2.246, 4120 6.120(x3), 2.048, 256, 8	0.001
Modelo 4	Genes 1 % tol. dif. (9.131)	ReLU	8	9.131, 8240 12.240(x3), 4.096, 512, 16	0.001
Modelo 5	Alelos 1 % tol. dif. (5.218)	ReLU	8	5.218, 8240 12.240(x3), 4.096, 512, 16	0.001

Ejecución modelo 1

```
Total genomes in project: 1632
Percentage of feature selection difference: 1%
Number of feature selection difference: 16
Total genes in project: 157611
Genes filtered and number of first layer: 9131
Total samples: 1618 samples
Train data: 1296 samples

100 pasadas - Tiempo de ejecución 38:16 min.

11/11 [==================] - 23s 1s/step - loss: 0.0967 - acc: 0.8812

Evaluate
3/3 [==================] - 1s 242ms/step - loss: 0.1974 - acc: 0.7488

Test data: 322 samples
loss: 0.19737181901931763 - acc: 0.7888198494911194


250 pasadas - Tiempo de ejecución 117:25 min.

11/11 [==================] - 23s 1s/step - loss: 0.0411 - acc: 0.9529

Evaluate
3/3 [==================] - 1s 238ms/step - loss: 0.1824 - acc: 0.7668
```

```
Test data: 322 samples
loss: 0.1824266004562378 - acc: 0.8167701959609985

500 pasadas - Tiempo de ejecución 297:57 min.

11/11 [==================] - 23s 1s/step - loss: 0.0130 - acc: 0.9877

Evaluate
3/3 [==================] - 1s 235ms/step - loss: 0.1814 - acc: 0.7668

Test data: 322 samples
loss: 0.1814118367433548 - acc: 0.8167701959609985
```

Ejecución modelo 2

```
Total genomes in project: 1632
Percentage of feature selection difference: 1%
Number of feature selection difference: 16
Total genes in project: 81957
Genes filtered and number of first layer: 5218
Total samples: 1618 samples
Train data: 1296 samples

100 pasadas - Tiempo de ejecución 28:24 min.

11/11 [==================] - 16s 1s/step - loss: 0.1011 - acc: 0.8765

Evaluate
3/3 [==================] - 1s 189ms/step - loss: 0.1730 - acc: 0.7578

Test data: 322 samples
loss: 0.17301113903522491 - acc: 0.7577639818191528

250 pasadas - Tiempo de ejecución 73:32min.

11/11 [==================] - 16s 957ms/step - loss: 0.0836 - acc: 0.8904

Evaluate
3/3 [==================] - 1s 175ms/step - loss: 0.1787 - acc: 0.7764

Test data: 322 samples
loss: 0.17263315618038177 - acc: 0.7764015624792056

500 pasadas - Tiempo de ejecución 147:47min.

11/11 [==================] - 16s 934ms/step - loss: 0.0437 - acc: 0.9421

Evaluate
3/3 [==================] - 1s 170ms/step - loss: 0.1726 - acc: 0.7795
```

```
Test data: 322 samples
loss: 0.17263315618038177 - acc: 0.7795031070709229
```

Ejecución modelo 3

```
Total genomes in project: 1632
Percentage of feature selection difference: 2%
Number of feature selection difference: 32
Total genes in project: 157611
Genes filtered and number of first layer: 2246
Total samples: 1618 samples
Train data: 1296 samples

100 pasadas - Tiempo de ejecución 9:43 min.

Epoch 100/100
11/11 [==================] - 3s 293ms/step - loss: 0.2127 - acc: 0.7353

Evaluate
3/3 [==================] - 0s 58ms/step - loss: 0.2217 - acc: 0.6708

Test data: 322 samples
loss: 0.22165624797344208 - acc: 0.6708074808120728

250 pasadas - Tiempo de ejecución 51:18 min.

Epoch 250/250
11/11 [==================] - 3s 268ms/step - loss: 0.1190 - acc: 0.8681

Evaluate
3/3 [==================] - 0s 50ms/step - loss: 0.1788 - acc: 0.7236

Test data: 322 samples
loss: 0.1788315325975418 - acc: 0.7236024737358093

500 pasadas - Tiempo de ejecución 255:57 mins.

Epoch 500/500
11/11 [==================] - 3s 280ms/step - loss: 0.0319 - acc: 0.9784

Evaluate
3/3 [==================] - 0s 51ms/step - loss: 0.1871 - acc: 0.7240

Test data: 322 samples
loss: 0.18714165687561035 - acc: 0.7639751434326172
```

Ejecución modelo 4

```
Total genomes in project: 1632
Percentage of feature selection difference: 1%
```

```
Number of feature selection difference: 16
Total genes in project: 157611
Genes filtered and number of first layer: 9131
Total samples: 1618 samples
Train data: 1296 samples

100 pasadas - Tiempo de ejecución 37:04 min.

Epoch 100/100
11/11 [==================] - 21s 1s/step - loss: 0.1867 - acc: 0.7886

Evaluate
3/3 [==================] - 1s 241ms/step - loss: 0.2029 - acc: 0.7298

Test data: 322 samples
loss: 0.20291215181350708 - acc: 0.7298136353492737

250 pasadas - Tiempo de ejecución 92:04 min.

11/11 [==================] - 21s 1s/step - loss: 0.1338 - acc: 0.8410

Evaluate
3/3 [==================] - 1s 212ms/step - loss: 0.1699 - acc: 0.7453

Test data: 322 samples
loss: 0.1698680967092514 - acc: 0.7453415989875793

500 pasadas - Tiempo de ejecución 197:43 min.

11/11 [==================] - 20s 1s/step - loss: 0.0486 - acc: 0.9637

Evaluate
3/3 [==================] - 1s 211ms/step - loss: 0.1585 - acc: 0.7578

Test data: 322 samples
loss: 0.15849840641021729 - acc: 0.7577639818191528
```

Ejecución modelo 5

```
Total genomes in project: 1632
Percentage of feature selection difference: 1%
Number of feature selection difference: 16
Total genes in project: 81957
Genes filtered and number of first layer: 5218
Total samples: 1618 samples
Train data: 1296 samples

100 pasadas - Tiempo de ejecución 20:15 min.
```

```
Epoch 100/100
11/11 [==================] - 13s 1s/step - loss: 0.1947 - acc: 0.7870

Evaluate
3/3 [==================] - 1s 257ms/step - loss: 0.2075 - acc: 0.7391

Test data: 322 samples
loss: 0.20748765766620636 - acc: 0.739130437374115

250 pasadas - Tiempo de ejecución 61:27 min.

Epoch 250/250
11/11 [==================] - 12s 1s/step - loss: 0.1421 - acc: 0.8248

Evaluate
3/3 [==================] - 1s 199ms/step - loss: 0.1763 - acc: 0.7547

Test data: 322 samples
loss: 0.17632612586021423 - acc: 0.7546584010124207

500 pasadas - Tiempo de ejecución 123:17 min.

Epoch 500/500
11/11 [==================] - 12s 1s/step - loss: 0.1003 - acc: 0.8858

Evaluate
3/3 [==================] - 1s 213ms/step - loss: 0.1437 - acc: 0.7871

Test data: 322 samples
loss: 0.14368228197097778 - acc: 0.7870807242393494
```

Model\Acc	**100 épocas**	**250 épocas**	**+500 épocas**	**tiempo de entr. +500 épocas**
Modelo 1	0.7488	0.7668	0.7668	297:57 min.
Modelo 2	0.7578	0.7764	0.7795	147:47min.
Modelo 3	0.6708	0.7236	0.7240	51:18 min.
Modelo 4	0.7298	0.7453	0.7578	197:43 min.
Modelo 5	0.7391	0.7547	0.7871	123:17 min.

Con las pruebas realizadas no se observó impacto significativo en el porcentaje de aciertos entre genes y alelos, viendo una reducción significativa en los tiempos de entrenamiento en estos ultimos (entre un 25 % y 50 % más rápidos). Por el ajuste de diferencia de muestras, un porcentaje de 1 % demostró tener los resultados más prometedores, mientras que valores mayores (Modelo 3, 2 %) devolvió los resultados de aciertos más bajos, aunque

su entrenamiento resultó en tiempos de ejecuciones menores a los demás modelos.

Capítulo 6

Desarrollo de la herramienta

6.1. Introducción

Habiendo creado y entrenado un modelo de inteligencia artificial que puede evaluar si el genoma de una muestra de una bacteria de *S. Pyogenes* es invasiva o no, se puede notar que los datos que recibe como entrada el modelo es una matriz de números 1 y 0, con un tamaño igual a la cantidad de genes que se encuentran en las muestras utilizadas en el entrenamiento, donde cada elemento de la matriz se vincula con un gen y tiene como valor 1 si la muestra tiene ese gen y 0 en caso contrario. El valor de salida del modelo es un valor binario, 1 si la muestra es invasiva y 0 si no lo es.

Si nos abstraemos de la parte biológica, se puede ver que los patrones que busca el modelo para que el resultado sea 1 o 0, se determinan según el valor que tiene cada elemento de una matriz de unos y ceros.

Entonces surge la pregunta, se puede utilizar el modelo para problemas con la misma estructura? Es decir problemas que respeten la estructura de los datos de entrada y los de salida. No importa la bacteria o el virus que se quiera tratar, mientras se pueda generar para cada muestra una matriz de pertenencia de los genes que se encuentran en todas las muestras, y no importa la pregunta binaria que se quiera preguntar, mientras que el valor de las etiquetas para cada muestra sea 1 o 0.

Resulta interesante desarrollar una herramienta que permita realizar un flujo como el que se explicó en el capítulo anterior, desde el procesamiento de los archivos utilizando programas de bioinformática ya existentes hasta el entrenamiento de un modelo y la obtención de las estadísticas del impacto de los genes en ese modelo, para cualquier bacteria o virus y cualquier pregunta binaria que se quiera evaluar, por medio de una interfaz gráfica amigable

para no informáticos.

6.2. Requerimientos

- Módulo de proyectos: un proyecto es el trabajo que se asocia a un organismo biológico, en el cual se pueden procesar los archivos correspondientes y crear los estudios deseados.
    - Creación (eligiendo el reino del organismo a investigar), edición y eliminación de proyectos.
    - Procesamiento de archivos FASTQ.
    - Procesamiento de archivos FASTA.
    - Procesamiento de archivos GBF y creación de matriz de pertenencia para cada muestra.
- Módulo de estudios: un estudio es la pregunta binaria que se quiere trabajar sobre un organismo, cada estudio tiene su red neuronal. Un mismo proyecto puede tener varios estudios y así investigar distintas propiedades biológicas sobre el mismo organismo.
    - Creación (eligiendo si usar alelos o no), edición y eliminación de estudios para un proyecto.
    - Generación de CSV con listado de las muestras pertenencientes al proyecto.
    - Carga de CSV con etiquetas para cada muestra.
    - Entrenamiento de red neuronal del estudio con las muestras que tienen etiqueta para ese estudio.
    - Evaluación de muestra nueva.
    - Análisis de modelo y obtención de estadísticas.
    - Generación de CSV con información para los genes de mayor impacto en el modelo.
    - Generación de archivo multifasta, alineamiento de archivo multifasta y IQtree para los genes de mayor impacto en el modelo.

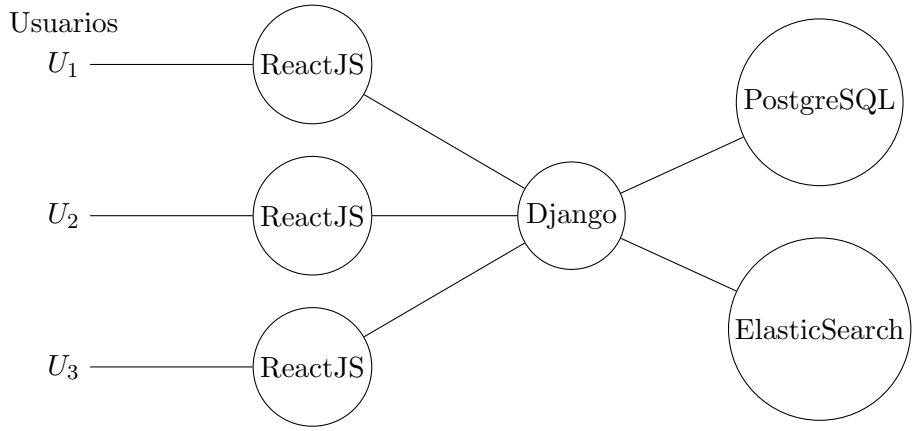
6.3. Arquitectura y tecnologías elegidas

6.3.1. Arquitectura

La herramienta se plantea como un sistema web. Una interfaz gráfica web que interactúa con una API REST, la cual dispone de dos bases de datos, una en PostgreSQL para el manejo de usuarios y sesiones y la otra no relacional en ElasticSearch para el almacenamiento de la gran cantidad de datos que implican los genomas. En total son cuatro servicios, y la arquitectura es la siguiente:

- Interfaz gráfica desarrollada con ReactJS.
- API RESTful desarrollada con Django.
- Base de datos en PostgreSQL.
- Base de datos no relacional en ElasticSearch.

Ejemplo de arquitectura con 3 usuarios usando la herramienta:



6.3.2. Tecnologías

- ReactJS es una librería para JavaScript para desarrollar SPA (single page applications), y la elección de la misma se debe a que es una librería simple de usar, eficiente y muy usada a nivel global. Las SPA son una buena opción cuando en la arquitectura de un sistema existe una API RESTful, ya que el usuario carga toda la interfaz con una sola petición al servidor web, y luego el resto de la interacciones se realizan

con la API RESTful para el intercambio de datos y el flujo de la aplicación. ReactJS además se puede integrar con Material Design, una librería creada por Google para diseñar interfaces gráficas siguiendo una línea de buenas prácticas de diseño.

- Django es un framework para Python. Más precisamente en este sistema se utiliza DRF (Django Rest Framework), un framework para desarrollar API's RESTful. También es simple de usar y uno de los frameworks más utilizados. Django cuenta con un ORM (object–relational mapping) para el manejo de bases de datos a alto nivel, con la opción de elegir entre varios motores de bases de datos, en este caso se eligió PostgreSQL para gestionar los usuarios y las sesiones.

- Como se observa en el capítulo anterior, el objetivo de este trabajo implica procesar y almacenar una gran cantidad de información, por eso resulta conveniente usar una base de datos no relacional como ElasticSearch, un motor de base de datos adecuado para realizar consultas sobre volúmenes de información con un tamaño grande como ocurre en este caso.

- Para la creación, entrenamiento y evaluación de las redes neuronales se utiliza la misma librería que en el capítulo anterior, Tensorflow para Python.

- Otra tecnología que se decide incorporar es Docker, la cual permite empaquetar una aplicación y sus dependencias en un contenedor. De esta manera, la herramienta se puede ejecutar en cualquier servidor que tenga instalado Docker.

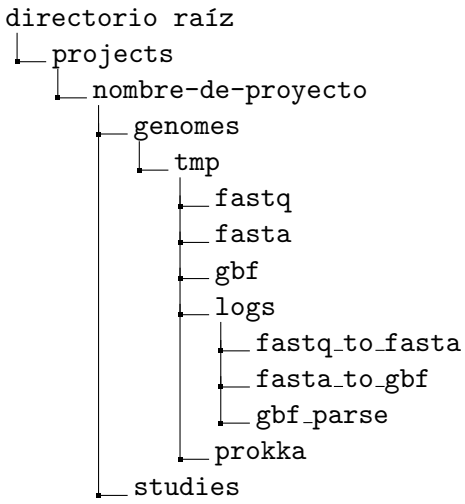
6.4. Desarrollo

6.4.1. Proyectos

Como se mencionó previamente, un proyecto es el espacio de trabajo que se asocia a un organismo biológico, el cual se crea para procesar los archivos FASTQ, FASTA y GBF vinculados al mismo, y también crear los estudios deseados.

Estructura de directorio

Al crear un nuevo proyecto, se agrega una carpeta con el nombre del proyecto en la carpeta **/projects**, la cual se encuentra en el directorio raíz de la herramienta. La carpeta creada se define con la estructura necesaria para el resto de las operaciones relacionadas al proyecto y los estudios que se creen. La estructura es la siguiente:



La carpeta **/genomes** es donde se agregan los archivos FASTQ, FASTA y GBF que se quieran procesar para el proyecto. Cuando se agregan archivos a un proyecto, la interfaz los detecta y permite acciones al usuario según el tipo de archivo y el estado del proyecto y sus estudios. Al igual que en el capítulo anterior el procesamiento de archivos FASTQ se hizo con *spades*, los archivos FASTA con *prokka* y los archivos GBF con un script propio, en la herramienta se procesan de la misma manera. La diferencia es que no se hace a través de una linea de comandos y con el requisito de saber cómo se usan esos programas, sino con un botón desde una interfaz gráfica y con una

herramienta que ya sabe cómo usar dichos programas y con qué parámetros, abstrayendo al usuario de ciertos conocimientos para que pueda concentrarse en la parte biológioca. Dentro de **/genomes** hay una carpeta **/tmp** que usa el sistema para el procesamiento y logs de cada tipo de archivo.

La carpeta **/studies** se crea para guardar la información necesaria de los estudios que se creen para el proyecto.

ElasticSearch

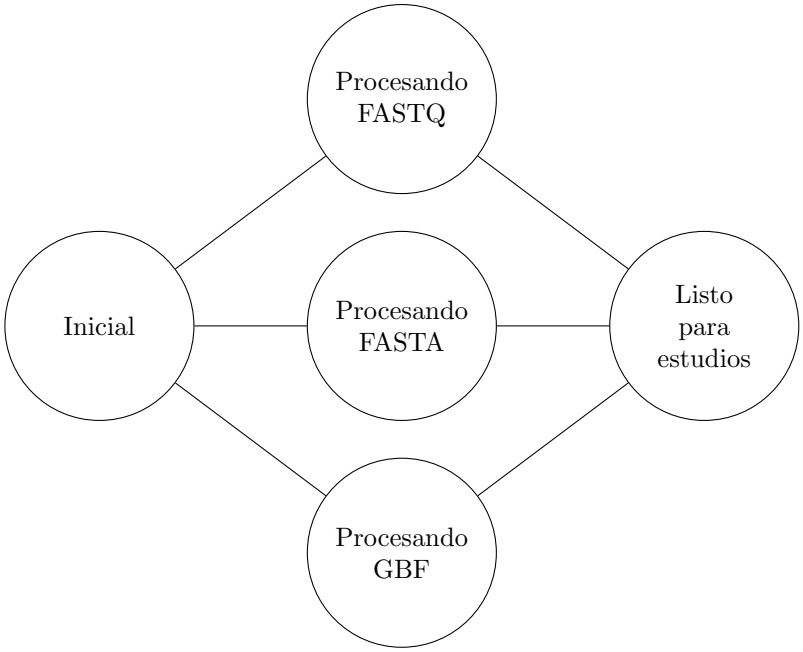
Además de la carpeta creada, también se crean en ElasticSearch (el servicio de base de datos no relacional para guardar la información de genomas y genes) los índices necesarios para el proyecto. Cada proyecto requiere tres índices, un índice para almacenar los genomas, uno para almacenar los genes y otro para almacenar los alelos:

1. Índice para genomas:
    - Se crea con el nombre: **nombre-de-proyecto-genomes-index**
    - En este índice se guardan los genomas que se procesan de los archivos GBF, persistiendo de cada uno el nombre de la muestra y el conjunto de genes que posee, donde por cada gen se dispone: nombre, producto, locus_tag y translation.
    - Al finalizar el procesamiento de un conjunto de GBFs, se crean para cada genoma las matrices de pertenencia de genes y la de alelos.
2. Índice para genes:
    - Se crea con el nombre: **nombre-de-proyecto-genes-index**
    - En este índice se guardan todos los genes que se obtienen de todos los genomas procesados, persistiendo para cada uno el nombre del gen, producto y translation (cadena de aminoácidos). Para distinguir un gen de otro se usa como atributo único su translation. A cada gen insertado se le asigna un identificador numérico, que luego se utiliza para crear las matrices de pertenencia de los genomas.
3. Índice para alelos:
    - Se crea con el nombre: **nombre-de-proyecto-alleles-index**

- Al igual que el índice de genes, en este índice se guardan todos los genes que se obtienen de todos los genomas procesados, pero en este caso los genes no se distinguen unos de otros por su cadena de aminoácidos, sino que se distinguen por su nombre, y se persiste para cada uno el nombre, producto, y los todos los translations que se encuentren para ese gen, es decir, los alelos. También se le asigna a cada gen insertado un identificador numérico, que luego se utiliza para crear las matrices de pertenencia de los genomas.

Estados del proyecto

El proyecto tiene 5 estados posibles:



- Inicial: es el estado en el cual se crea el proyecto, y es el estado al que vuelve el proyecto luego de procesar archivos si el estado previo al procesamiento era **Inicial**.
- Procesando FASTQ: es el estado al que pasa el proyecto cuando se procesan archivos FASTQ, para poder procesar archivos FASTQ el estado del proyecto debe ser **Inicial** o **Listo para estudios**. Al terminar

el procesamiento y convertir los archivos FASTQ a FASTA, vuelve al estado previo del procesamiento.

- Procesando FASTA: es el estado al que pasa el proyecto cuando se procesan archivos FASTA, para poder procesar archivos FASTA el estado del proyecto debe ser **Inicial** o **Listo para estudios**. Al terminar el procesamiento y convertir los archivos FASTA a GBF, vuelve al estado previo del procesamiento.
- Procesando GBF: es el estado al que pasa el proyecto cuando se procesan archivos GBF, para poder procesar archivos GBF el estado del proyecto debe ser **Inicial** o **Listo para estudios**. Al terminar el procesamiento de los archivos GBF y crear las matrices de pertenencia de los genomas, pasa al estado **Listo para estudios**.
- Listo para estudios: es el estado al que pasa el proyecto luego de procesar archivos GBF, y es el estado que habilita acciones sobre los estudios del proyecto.

Procesamiento de archivos

Esta herramienta requiere el procesamiento de tres tipos de archivo: el de archivos FASTQ, que se realiza con el programa *spades*, quien convierte los archivos a FASTA. El procesamiento de archivos FASTA, que se hace con *prokka*, convirtiendo los mismos a GBF. Y por último, el procesamiento de archivos GBF, que funciona de la siguiente manera:

1. Por cada archivo GBF que se encuentre en la carpeta del proyecto:
    a. Se crea un JSON con el nombre de la muestra y una lista vacía para ir agregando los genes.
    b. Por cada gen que se encuentra en el archivo se lo agrega a la lista guardando: su nombre, el producto, locus_tag y translation (cadena de aminoácidos del gen).
    c. Cuando se termina de procesar el archivo, se guarda el JSON en la base de datos ElasticSearch, en el índice de genomas del proyecto. A continuación se puede ver un ejemplo:

```
{
    "name": "ERR227094"
    "genes" : [
        {
            "product" : "Maltodextrin phosphorylase",
            "gene" : "malP",
            "translation" : "MTRFTEYVETKLGKSLTQAS...",
            "locus_tag" : "JJOEMFFN_00006"
        },
        {
            "product" : "hypothetical protein",
            "gene" : "",
            "translation" : "MTKKHLLTLLLISFFTSFLV...",
            "locus_tag" : "JJOEMFFN_00007"
        },
        ...
    ]
}
```

Extracto 3: Ejemplo de JSON para un genoma obtenido de un GBF

2. Luego de insertar los nuevos genomas, se obtienen todos los genomas del proyecto (los recién insertados y los ya existentes) para obtener todos los genes y armar dos conjuntos de genes:

    a. Un conjunto eligiendo como atributo único la cadena de aminoácidos del gen, cuyos genes se insertan en el índice de genes del proyecto. A cada gen se le agrega un atributo id numérico que se utiliza para identificar el gen dentro del índice. Ejemplo de un JSON creado para un gen:

```
{
    "gene" : "truA",
    "product" : "tRNA pseudouridine synthase A",
    "translation" : "MVRYKATISYDGTLFSGFQRQR...",
    "id" : 1
}
```

Extracto 4: Ejemplo de JSON para un gen obtenido de un GBF

    b. Otro conjunto eligiendo como atributo único el nombre del gen, para agrupar en este caso las distintas cadenas de aminoácidos

posibles (alelos) para cada gen, cuyos genes se insertan en el índice de alelos del proyecto. A cada gen se le agrega un atributo id numérico que se utiliza para identificar el gen dentro del índice. Ejemplo de un JSON creado para un gen:

```
{
    "gene" : "truA",
    "product" : "tRNA pseudouridine synthase A",
    "translation" : [
        "MVRYKATISYDGTLFSGFQRQRHLRTVQEEIEKTLYKLNNGTKII...",
        "MVRYKATISYDGTLFSGFQRQRHLRTVQEEIEKTLYKLNNGTKIM...",
        "MVRYKATISYDGTLFSGFQRQRHLRTVQEEIEKTLYKLNNSTKII...",
        ...
    ],
    "id" : 1
}
```

Extracto 5: Ejemplo de JSON para un gen con alelos obtenido de un GBF

3. Una vez creados los índices de genes y de alelos, se pueden generar las matrices de pertenencia de los genomas, donde para cada genoma se crean dos matrices: la matriz de pertenencia de genes y la matriz de pertenencia de alelos. El objetivo es agregar a estas matrices los identificadores de los genes que contenga el genoma. Cada matriz se corresponde con un índice distinto, en la matriz de genes se agregan los identificadores de los genes del índice de genes y en la matriz de alelos los identificadores del índice de alelos. Debajo se muestra un ejemplo de un genoma luego de crear las matrices de pertenencia:

```
{
    "name": "ERR227094"
    "genes" : [
        {
            "product" : "Maltodextrin phosphorylase",
            "gene" : "malP",
            "translation" : "MTRFTEYVETKLGKSLTQAS...",
            "locus_tag" : "JJOEMFFN_00006"
        },
        {
            "product" : "hypothetical protein",
            "gene" : "",
            "translation" : "MTKKHLLTLLLISFFTSFLV...",
            "locus_tag" : "JJOEMFFN_00007"
        },
        ...
    ],
    "gen_pertenence" : [258, 1762, 1763, ...],
    "alleles_pertenence" : [258, 1659, 1660, ...]
}
```

Extracto 6: Ejemplo de JSON para un genoma con sus matrices de pertenencia

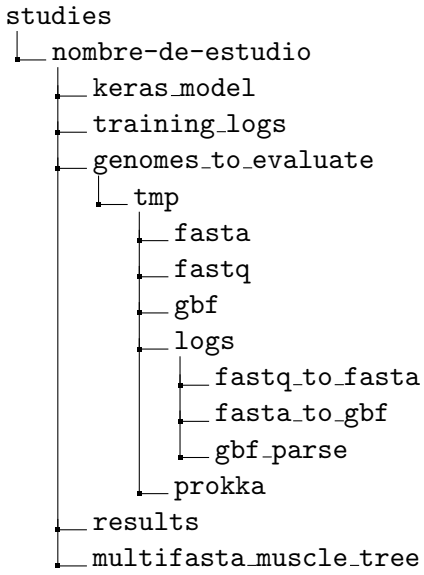
6.4.2. Estudios

Como se mencionó previamente, un estudio es la pregunta binaria que se quiere trabajar sobre un organismo, por ejemplo, aplicando el mismo caso descripto en el capítulo anterior: si el objetivo es estudiar la invasividad de la bacteria *S. pyogenes*, se crea un proyecto para dicha bacteria y dentro del mismo se crea el estudio **Invasividad**. Al momento de crear un estudio se debe elegir si se desea trabajar con genes o alelos.

El propósito de crear un estudio es el de entrenar una red neuronal con las muestras obtenidas de la bacteria a tratar, para que detecte los patrones que determinan el resultado de la pregunta binaria que se quiere investigar.

Estructura de directorio

Al crear un nuevo estudio, se agrega una carpeta con el nombre del estudio en la carpeta **/studies** del proyecto. La carpeta creada se define con la estructura necesaria para el resto de las operaciones relacionadas al estudio. La estructura es la siguiente:



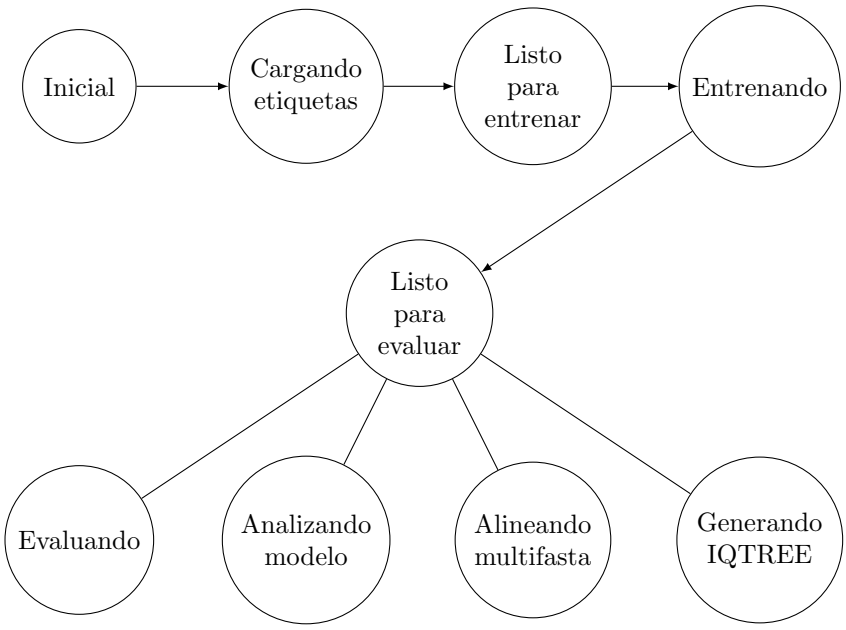
La carpeta **/keras_model** se utiliza para guardar el modelo y toda la estructura de archivos y carpetas que requiere Keras para entrenar la red neuronal. En **/training_logs** se almacenan todos los logs de los entrenamientos y optimizaciones que se realizan sobre el modelo, además de los análisis que se hagan sobre el modelo con la librería Shap.

La carpeta **/genomes_to_evaluate** es donde se agregan los archivos FASTQ, FASTA o GBF que se quieran evaluar con la red neuronal, y dentro de la misma se encuentran los subdirectorios necesarios para el procesamiento de los archivos de la misma manera que ocurre en el proyecto. En la carpeta **/results** se guardan los resultados de las evaluaciones de los genomas por parte de la red neuronal.

Por último, **/multifasta_muscle_tree** se usa al crear archivos multifasta, para alinear archivos multifasta y para generar archivos IQTree.

Estados del estudio

El estudio tiene 9 estados posibles:



- Inicial: es el estado en el cual se crea el estudio.
- Cargando etiquetas: es el estado al que pasa el estudio cuando se cargan etiquetas. Para poder cargar etiquetas el estado del estudio debe ser **Inicial**, **Listo para entrenar** o **Listo para evaluar**. Al terminar de etiquetar los genomas, el estudio pasa al estado **Listo para entrenar**.
- Listo para entrenar: es el estado al que pasa el estudio luego de cargar etiquetas, y es el estado que habilita el entrenamiento de la red neuronal.
- Entrenando: es el estado al que pasa el estudio al entrenar la red neuronal. Para poder entrenar el estado del estudio debe ser **Listo para entrenar** o **Listo para evaluar**. Al terminar el entrenamiento el estudio pasa al estado **Listo para evaluar**.
- Listo para evaluar: es el estado al que pasa el estudio luego de entrenar la red neuronal, y es el estado que habilita las acciones para evaluar un

genoma nuevo, analizar el modelo para obtener estadísticas, generar y alinear multifastas y generar IQTREE.

- Evaluando: es el estado al que pasa el estudio al evaluar un genoma nuevo.
- Analizando modelo: es el estado al que pasa el estudio al analizar un modelo entrenado. Al terminar el análsis y detectar los genes de mayor impacto se permite generar los multifasta, alinearlos, generar IQTREE y el detalle para un gen.
- Alineando multifasta: es el estado al que pasa el estudio mientras se alinea un multifasta.
- Generando IQTREE: es el estado al que pasa el estudio mientras se genera un IQtree.

Entrenamiento de redes neuronales y evaluación

Con el estudio creado, lo que se necesita para poder crear la red neuronal y empezar con el entrenamiento es etiquetar los genomas del proyecto al que pertenece el estudio.

Para etiquetar las muestras, la herramienta permite exportar un CSV que cuenta con dos columnas: una columna con el nombre de cada muestra perteneciente al proyecto y la otra es una columna vacía para completar con el valor de cada etiqueta. Luego este CSV se puede cargar para que las muestras sean etiquetadas.

Lo que implica esta carga de etiquetas es agregar un atributo con el nombre del estudio y el valor de la etiqueta, en cada genoma indicado en el CSV en el índice de genomas del proyecto. Entonces, el JSON que representa a un genoma en ElasticSearch, luego de agregar la etiqueta para el estudio **Invasividad** con etiqueta **1**, queda asi:

```
{
    "name": "ERR227094"
    "genes" : [
        {
            "product" : "Maltodextrin phosphorylase",
            "gene" : "malP",
            "translation" : "MTRFTEYVETKLGKSLTQAS...",
            "locus_tag" : "JJOEMFFN_00006"
        },
        {
            "product" : "hypothetical protein",
            "gene" : "",
            "translation" : "MTKKHLLTLLLISFFTSFLV...",
            "locus_tag" : "JJOEMFFN_00007"
        },
        ...
    ],
    "gen_pertenence" : [258, 1762, 1763, ...],
    "alleles_pertenence" : [258, 1659, 1660, ...],
    "invasividad": 1
}
```

Extracto 7: Ejemplo de JSON para un genoma etiquetado

Etiquetar las muestras habilita el entrenamiento de la red neuronal. El entrenamiento admite varios parámetros para realizar distintas pruebas y llegar a la combinación que genere los mejores resultados de tasa de acierto y error. Los parámetros que se pueden especificar son los siguientes:

- Cantidad de pasadas: define la cantidad de pasadas a realizar en el entrenamiento con el conjunto de datos existente.
- Porcentaje de tolerancia de diferencia entre conjuntos de etiquetas: de igual manera que se explicó en el capítulo anterior, este define los genes que se deben tener en cuenta en la generación de los datos de entrada.
- Porcentaje de datos para entrenar (el resto se usa para testear el aprendizaje): define la cantidad de muestras que deben usarse para el entrenamiento y el testeo.
- Learning rate: define el learning rate del entrenamiento de la red.

- Arquitectura: define la arquitectura de la red, más precisamente las capas ocultas de la red, ya que la capa de entrada siempre es la cantidad de genes filtrados y la capa de salida es una capa con una sola neurona. La capa de entrada y las ocultas tienen definido como función de activación la función ReLU, y la capa de salida la función sigmoide.

Con estos parámetros definidos, la herramienta tiene todo definido para realizar el entrenamiento. El flujo es el siguiente:

1. Obtener todos los genomas del proyecto que poseen una etiqueta para el estudio.
2. Para cada gen de dichos genomas, determinar a cuántos genomas etiquetados con 0 pertenece y a cuántos con 1.
3. Filtrar los genes para descartar los que no cumplan con el porcentaje de tolerancia de diferencia definido previamente (si las cantidades entre los genomas etiquetados con 1 y 0 al cual pertenece un gen es menor al parámetro establecido, el gen se descarta).
4. Transformar las matrices de pertenencia de los genomas (listas de ids de genes que contiene un genoma y que pueden variar en su longitud para cada genoma) en listas de igual longitud para todos los genomas para que sean los datos de entrada de la red neuronal. La transformación es la siguiente: para cada genoma se arma una lista con la longitud de genes que quedaron filtrados previamente, donde cada posición de la lista representa uno de esos genes filtrados, y que cada genoma puede poseer o no, si el genoma posee el gen esa posición tendrá un 1 y en su defecto un 0.
5. En este punto hay para cada genoma del proyecto etiquetado para el estudio una matriz de longitud igual a la cantidad de genes filtrados con valores posibles 0 y 1 según los genes que contenga el genoma.
6. Con los datos ya preparados, el próximo paso es crear la red con la arquitectura.
7. Por último queda comenzar el entrenamiento del modelo separando los datos para entrenamiento y testeo según el porcentaje definido, y con el learning rate y la cantidad de pasadas establecidos.

Una vez que termina el primer entrenamiento de la red de un estudio, se puede seguir optimizando el modelo, es decir, realizar más pasadas con

los mismos valores que se habían definido para los parámetros en el entrenamiento, para que la misma mejore su tasa de acierto y reduzca el error.

Por otro lado, también es posible en un estudio volver a realizar un entrenamiento con otros valores en los parámetros, pero esto implica crear una nueva red y perder la anterior.

Estadísticas pos entrenamiento e impacto de genes

La última sección que tiene un estudio es el análisis del modelo entrenado. Algo interesante de tener una red neuronal que reconoce qué patrones hacen que un genoma dé un resultado u otro, que para el ejemplo en cuestión sería invasiva o no, es poder ver cuáles son esos patrones, es decir, cuáles son las neuronas (genes) o combinaciones de neuronas que tienen mayor impacto en el resultado.

Luego de realizar un primer entrenamiento para un modelo de un estudio, se habilita la acción para analizar dicho modelo. Este análisis es un proceso que se lleva a cabo utilizando la librería **SHAP (SHapley Additive exPlanations)**, una librería para Python que se usa para explicar las predicciones de modelos de *Machine Learning*. La misma contiene funciones integradas que hacen uso del método **Shapley value**[10], el cual pretende determinar cuál es el aporte que brinda cada participante de un grupo a un resultado final.

El funcionamiento de este método consiste en obtener el Shapley value para cada feature de los datos de entrada de un modelo (en este caso, para cada gen de los datos de entrada) y así determinar qué impacto tiene cada gen sobre las predicciones del modelo.

Los pasos para calcular el Shapley value para un gen son los siguientes:

1. Crear un conjunto con todas las combinaciones posibles de features para calcular el valor promedio de predicción del modelo.
2. Crear un conjunto con todas las combinaciones posibles de features ignorando el gen a analizar.
3. Para cada combinación del paso 2:
    a. Calcular la diferencia entre el valor de la predicción del modelo sin el gen (con un 0 en la posición de dicho gen) y el valor promedio.
    b. Calcular la diferencia entre el valor de la predicción del modelo con el gen (con un 1 en la posición de dicho gen) y el valor promedio.

c. Calcular la diferencia de los valores de los pasos **b.** y **a.** para calcular cuánto afecta el gen en dicha combinación al valor promedio.

4. Obtener el promedio de todos los valores calculados en el paso **3.c.** para determinar el aporte promedio del gen, es decir, el Shapley value del gen.

En resumen, el Shapley value de un gen es el aporte promedio que el gen brinda al modelo a través de todas las posibles combinaciones de genes.

Luego de obtener el Shapley value de cada gen, el análisis se queda con los 50 genes con mayor impacto y de esa forma genera en la interfaz:

- Un gráfico donde se aprecia el impacto de cada uno de los 50 genes.
- Un listado para detallar de cada gen su id, su nombre, su product, la cantidad de genomas en cuales está presente, cuántos de esos genomas están etiquetados con SI y cuántos con NO, y por último el conjunto de translations (alelos) obtenidos para ese gen. Además por cada gen del listado se permiten cuatro acciones.

Las cuatro acciones que se pueden ejecutar son las siguientes:

- Generar multifasta: genera un archivo que contiene todos los translations (alelos) del gen en un formato FASTA. Esto quiere decir, que por cada translation crea dos líneas: la primera como cabecera, la cual empieza con '>', proporciona un nombre/identificador único a la secuencia, y en la segunda el translation.
- Alinear multifasta: sirve para alinear un archivo multifasta, o sea todos los translations que contiene un multifasta, para comparar las diferencias entre los aminoácidos de cada cadena.
- Generar IQtree: genera un archivo para ser visualizado por el programa IQtree, este mismo se genera a partir del alineamiento del multifasta. Con el IQtree se puede observar un árbol que explica los distintos conjuntos en los cuales se clasifican los translations de un gen.
- Detalle del gen en XLSX: genera un archivo Excel con dos hojas:

    1. Un listado de los translations del gen, donde por cada uno detalla en cuántos genomas está presente y cuántos de ellos fueron etiquetados con SI y cuántos con NO.

2. Un listado de los genomas al cual pertenece al gen, donde por cada uno se detalla la etiqueta de ese genoma y el translation del gen que contiene dicho genoma.

Con estas cuatro acciones disponibles para los 50 genes de mayor impacto en las predicciones de una red neuronal, los usuarios pueden obtener información más detallada y estadísticas sobre genes que resultan de mayor interés sobre el resto de los genes, y que si lo requieren, lleguen a un análisis en laboratorio.

Capítulo 7

Flujo de la herramienta

7.1. Flujo y uso

En esta sección se mostrará el flujo y uso general de la herramienta, visualizando imágenes de la interfaz gráfica para un mejor entendimiento, y tomando como ejemplo el caso de uso descripto en el capítulo anterior.

7.1.1. Proyectos

Listado de proyectos

El punto inicial del flujo de trabajo, es decir la pantalla principal, es el módulo de proyectos, en el que se muestra una tabla con los proyectos existentes. En la imagen a continuación se puede ver dicha tabla previa creación de algún proyecto.

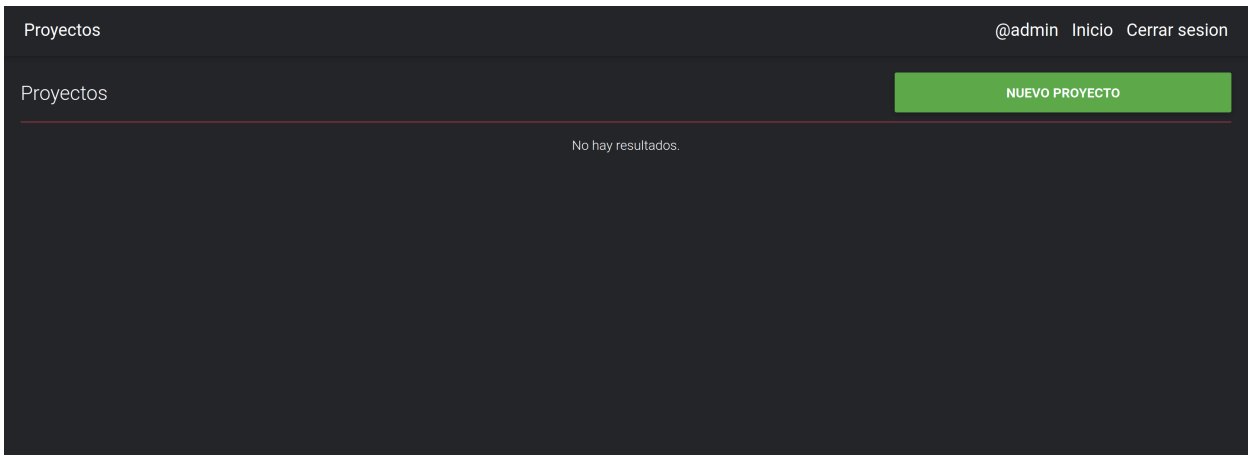


Figura 7.1: Listado de proyectos

Creación de proyecto

Para crear un proyecto, se dispone un formulario que se abre mediante el botón **Nuevo proyecto**. En el formulario hay tres campos a completar con nombre, descripción y reino deseados. El nombre debe ser único entre los proyectos existentes, la descripción es opcional y el reino determina cómo los procesos bioinformáticos deben procesar los archivos que representan los genomas. En las tres imágenes que continuan se ve el formulario durante la creación y luego la respuesta exitosa del proyecto creado.

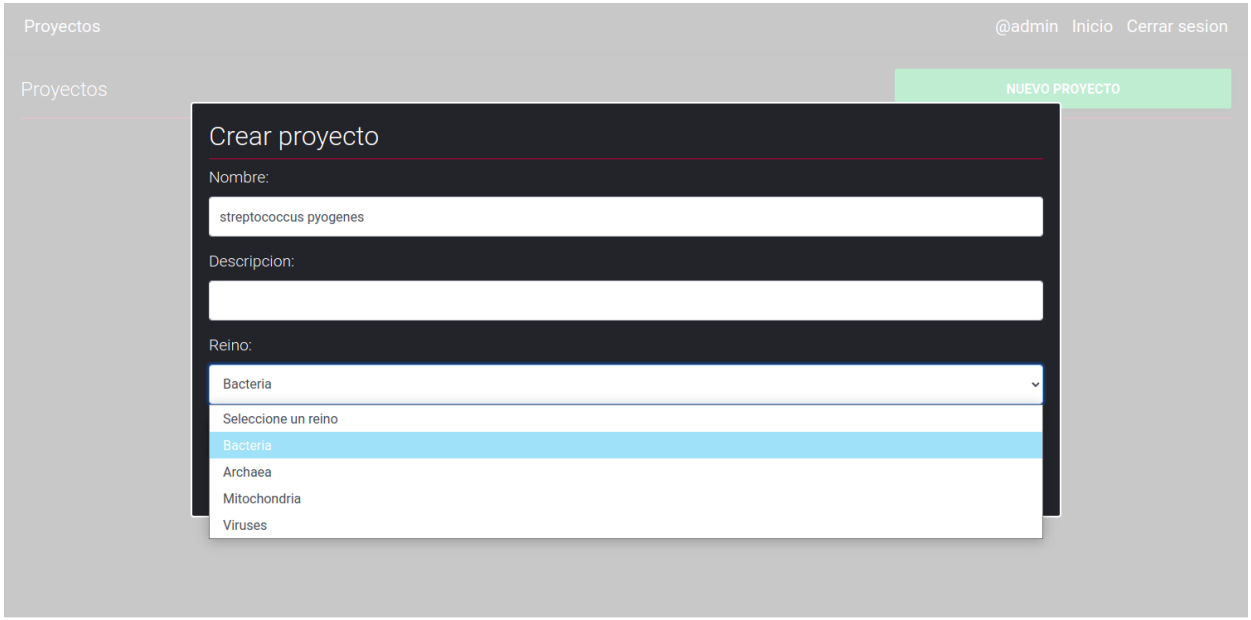


Figura 7.2: Creación de proyecto - Completando formulario

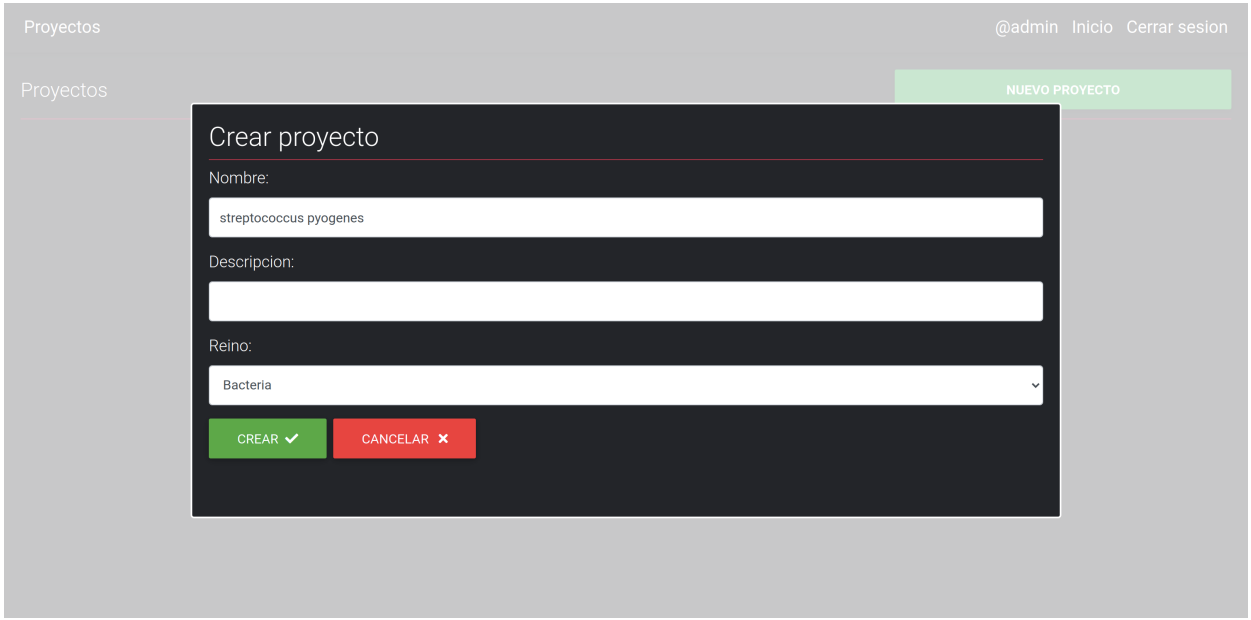


Figura 7.3: Creación de proyecto - Formulario completo

Figura 7.4: Creación de proyecto - Proyecto creado

Luego de crear un proyecto, el mismo aparecerá en el listado de proyectos, visualizando su nombre, la carpeta creada para el mismo, el reino, la cantidad de genomas, de genes y de alelos, la cantidad de genomas con matriz de pertenencia, el estado del proyecto, la cantidad de archivos FASTQ, FASTA y GBF sin procesar, y algunas acciones para editar, eliminar o acceder al proyecto.

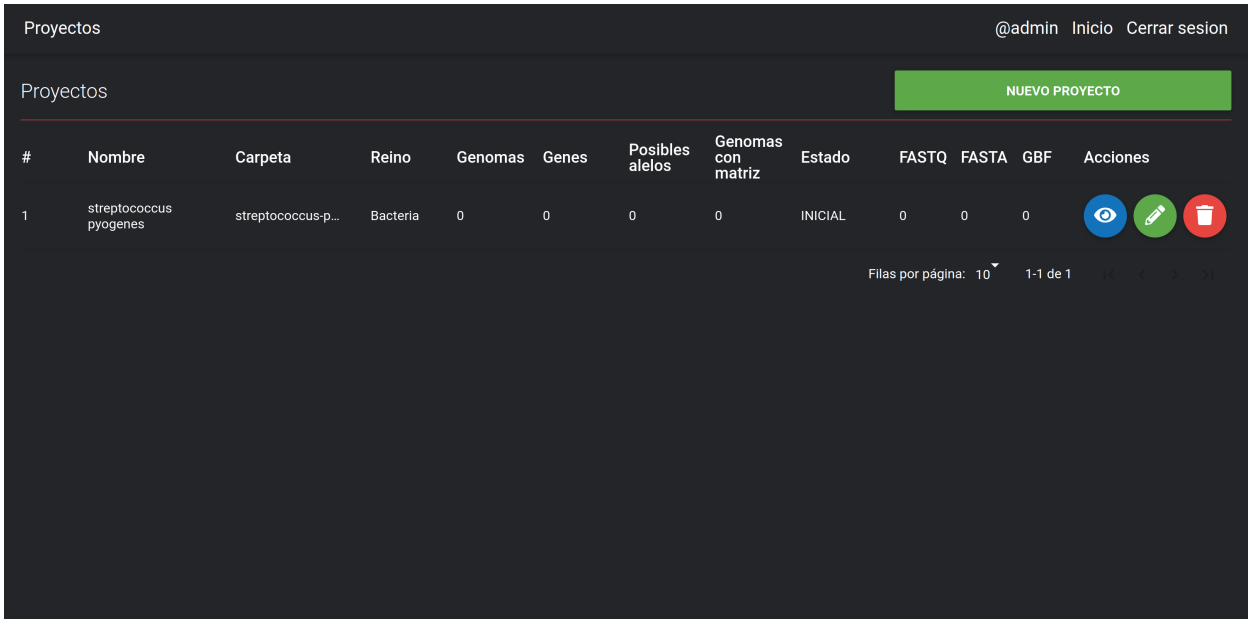


Figura 7.5: Listado de proyectos

Edición de proyecto

Para un proyecto existente se puede editar la descripción abriendo el formulario de edición desde la tabla de proyectos. En las dos imágenes a continuación se muestra la edición de un proyecto.

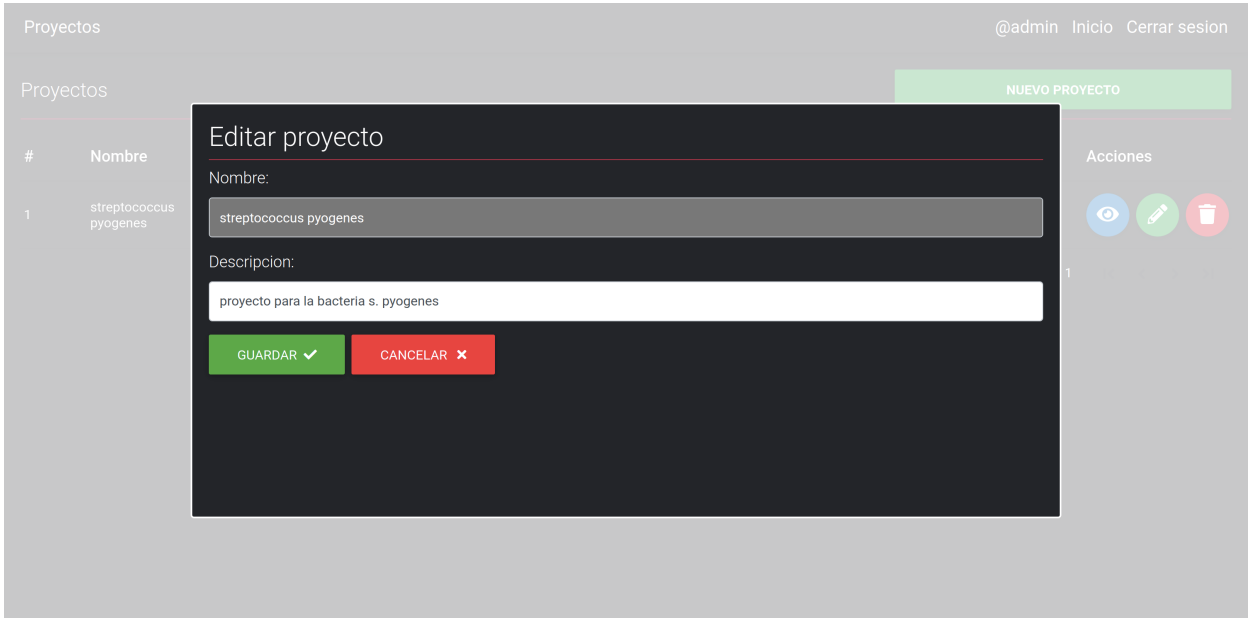


Figura 7.6: Edición de proyecto - Editando descripción

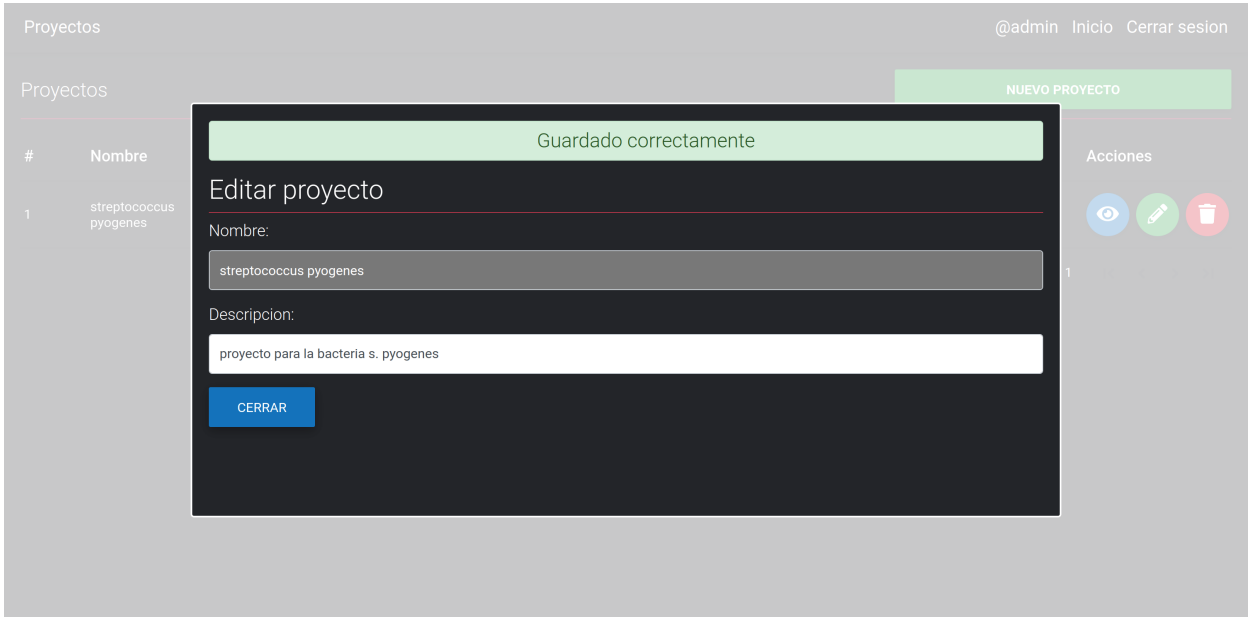


Figura 7.7: Edición de proyecto - Proyecto editado

Eliminación de proyecto

Un proyecto existente se puede eliminar, esto implica eliminar todos los estudios que contenga, los índices en ElasticSearch (genomas, genes y alelos) y las carpetas creadas para el mismo (modelos creados, logs, archivos, etc.). Es una operación que no puede deshacerse. En la siguiente imagen se ve la eliminación de un proyecto.

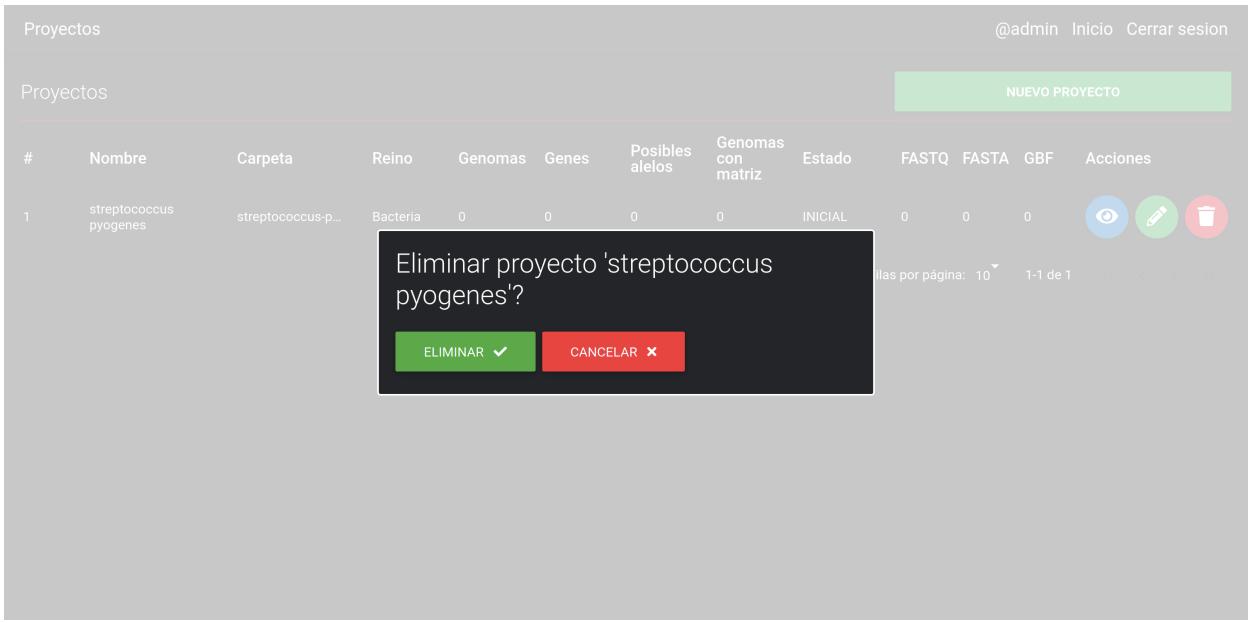


Figura 7.8: Eliminación de proyecto

7.1.2. Detalle de proyecto y listado de estudios

El módulo que continúa el flujo de trabajo luego de crear un proyecto es el del detalle de un proyecto, al cual se puede acceder mediante un botón disponible para cada proyecto existente en el listado de proyectos. En este módulo se ve más información sobre un proyecto, es donde se pueden realizar todas las acciones disponibles para el mismo, y también donde se pueden crear estudios y ver los que ya posee.

Detalle de proyecto

En la imagen siguiente se visualiza un ejemplo del detalle de un proyecto. En la parte superior se muestra el nombre y la acción que se puede realizar según el estado del proyecto, en este caso no hay archivos FASTQ, FASTA ni GBF para procesar, por eso no hay ninguna acción posible. Debajo hay campos de texto que muestran el estado, la descripción, el reino, la cantidad de archivos FASTQ, FASTA y GBF disponibles para procesar, la cantidad de genomas insertados en el índice de genomas, la cantidad de genes totales, la cantidad de alelos y la cantidad de genomas que tienen matriz de pertenencia.

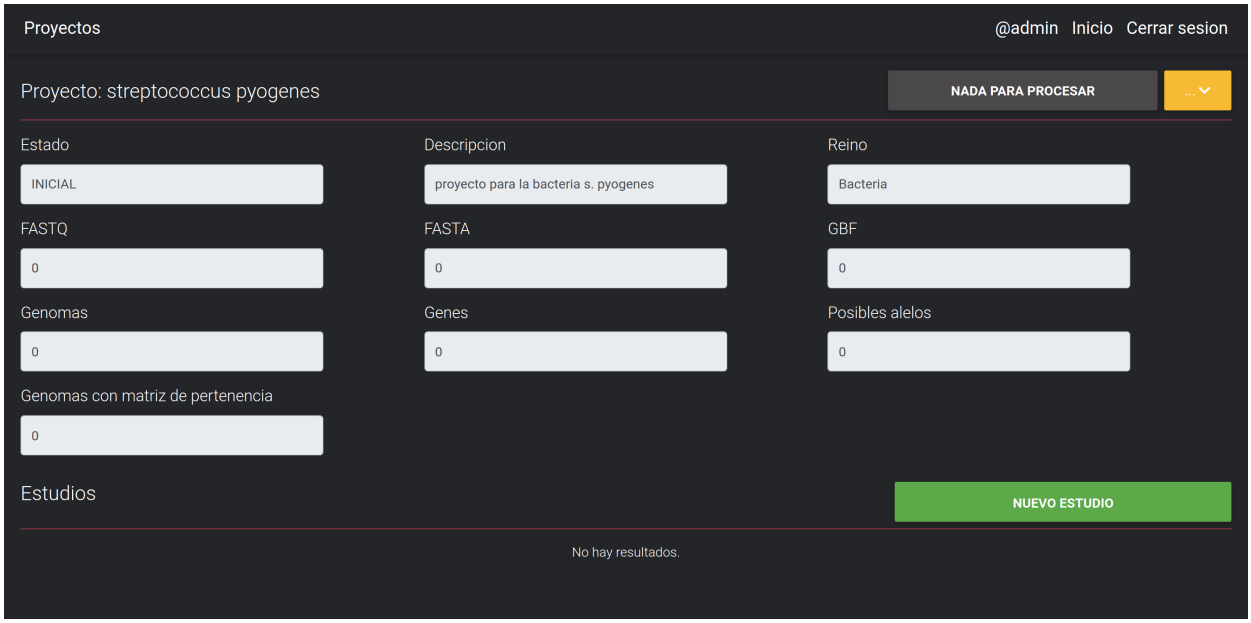


Figura 7.9: Detalle de proyecto - Inicial

Para procesar los 1638 archivos FASTQ mencionados en el capítulo anterior, se deben agregar los mismos en la carpeta del proyecto correspondiente, luego la herramienta los detecta y muestra la cantidad detectada. Y si el estado del proyecto lo permite, se habilita una acción en la parte superior para procesar dichos archivos FASTQ.

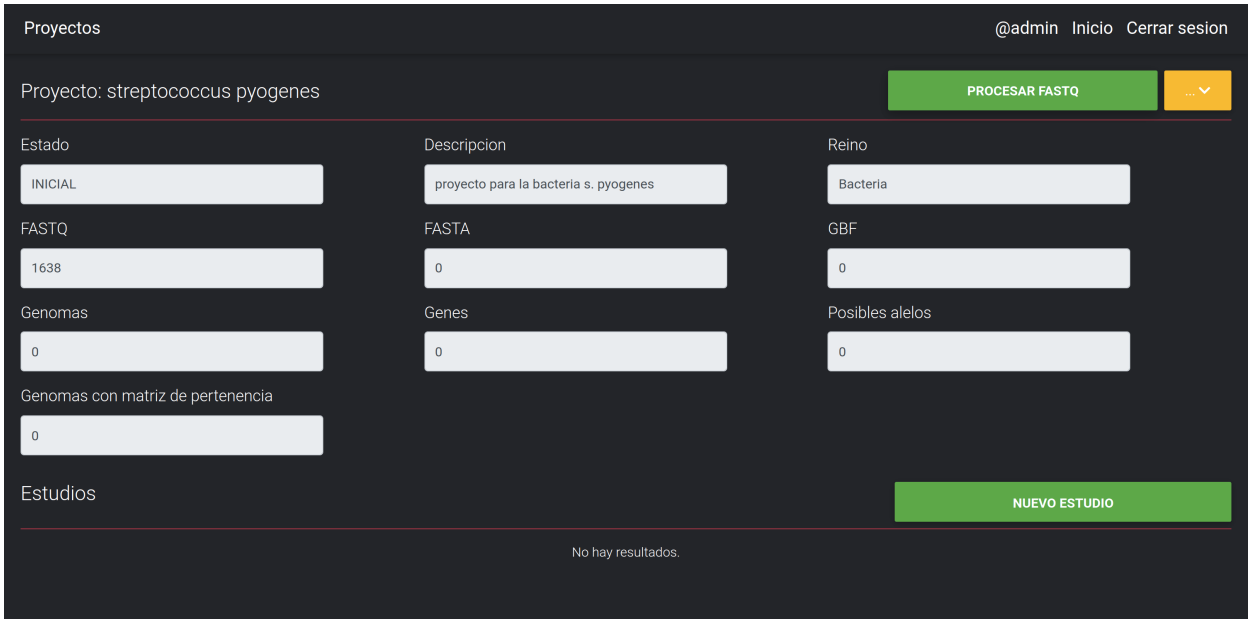


Figura 7.10: Detalle de proyecto - Procesar FASTQ

Al hacer click en **Procesar FASTQ** primero se debe ingresar la cantidad de CPUs que se desean usar para procesar los archivos. El número máximo de CPUs está definido por la variable de entorno al momento de configurar la herramienta.

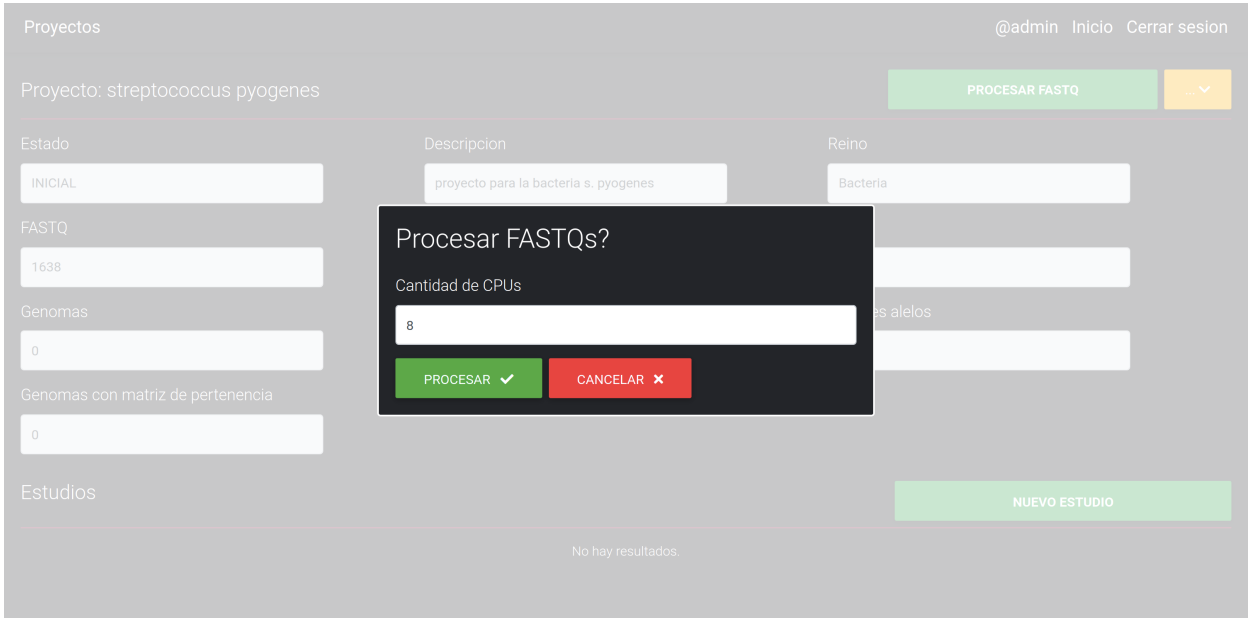


Figura 7.11: Detalle de proyecto - Procesar FASTQ

Luego de indicar la cantidad de CPUs y mandar a procesar, el estado del proyecto cambia a **Procesando FASTQ**. Esto inhabilita otras acciones como procesar FASTA y GBF o acciones dentro de un estudio, pero si se pueden dar de alta estudios y editarlos.

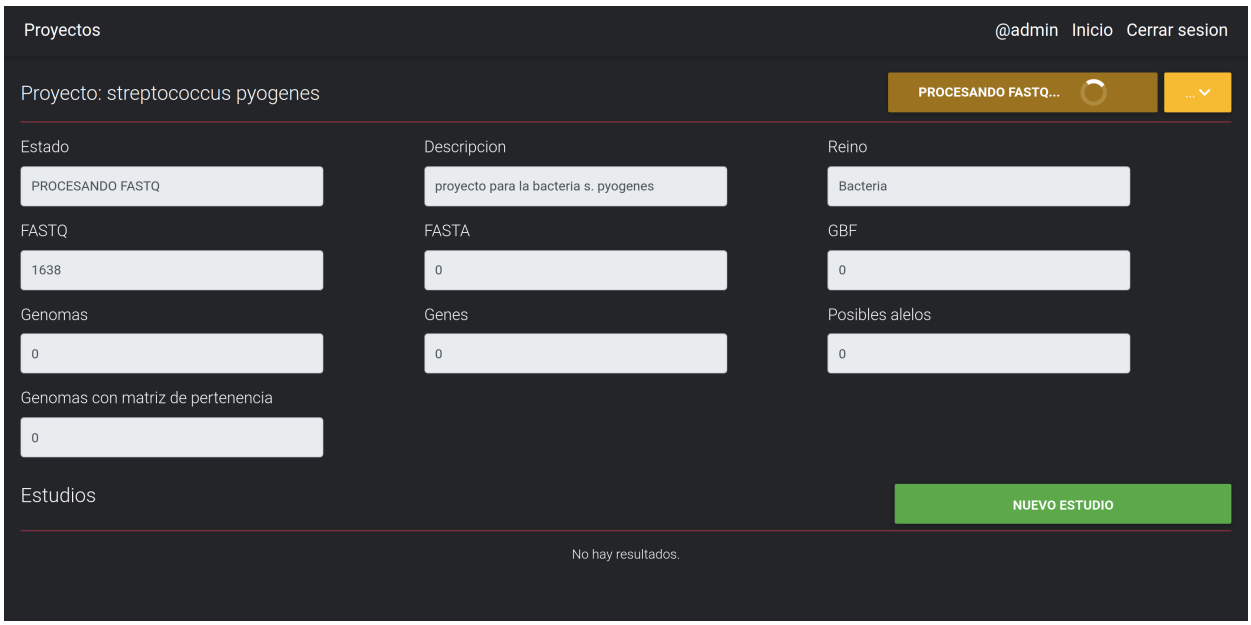


Figura 7.12: Detalle de proyecto - Procesando FASTQ

Al terminar el procesamiento de los archivos FASTQ y convertirlos a FASTA, el estado del proyecto cambia a **Inicial** y se puede ver que el campo de FASTQ quedó en 0 y ahora hay 1638 archivos FASTA listos para procesar, habilitando la acción **Procesar FASTA**. El procesamiento de archivos FASTA es similar al de los archivos FASTQ, a diferencia de que el estado cambia a **Procesando FASTA**.

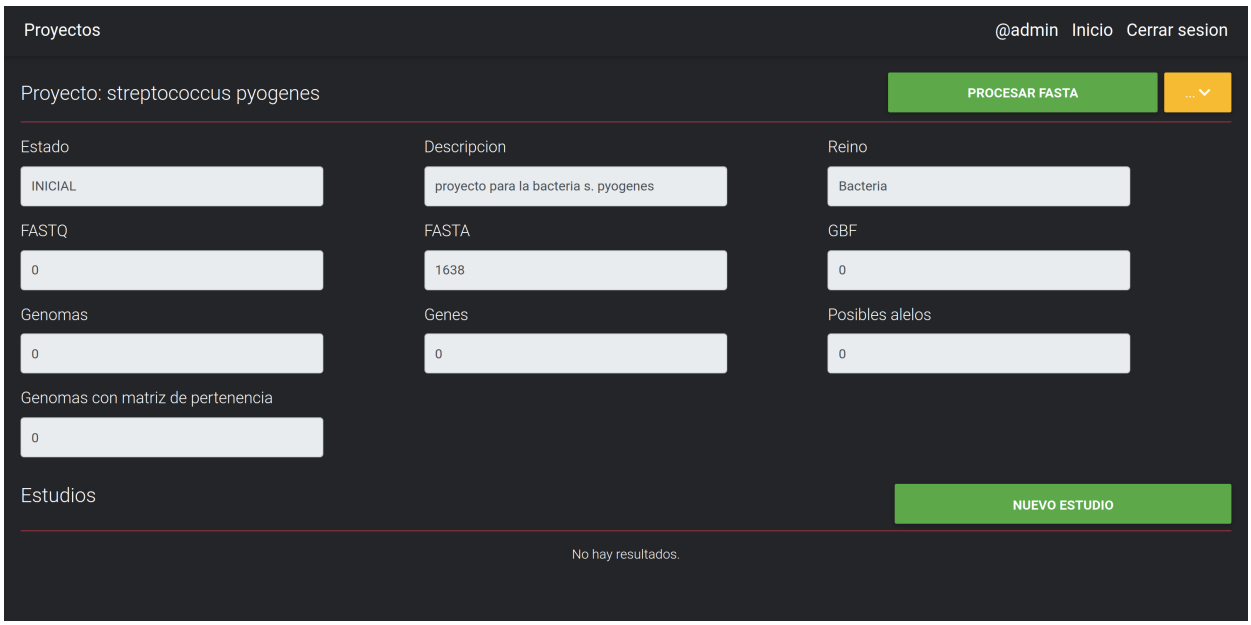


Figura 7.13: Detalle de proyecto - Procesar FASTA

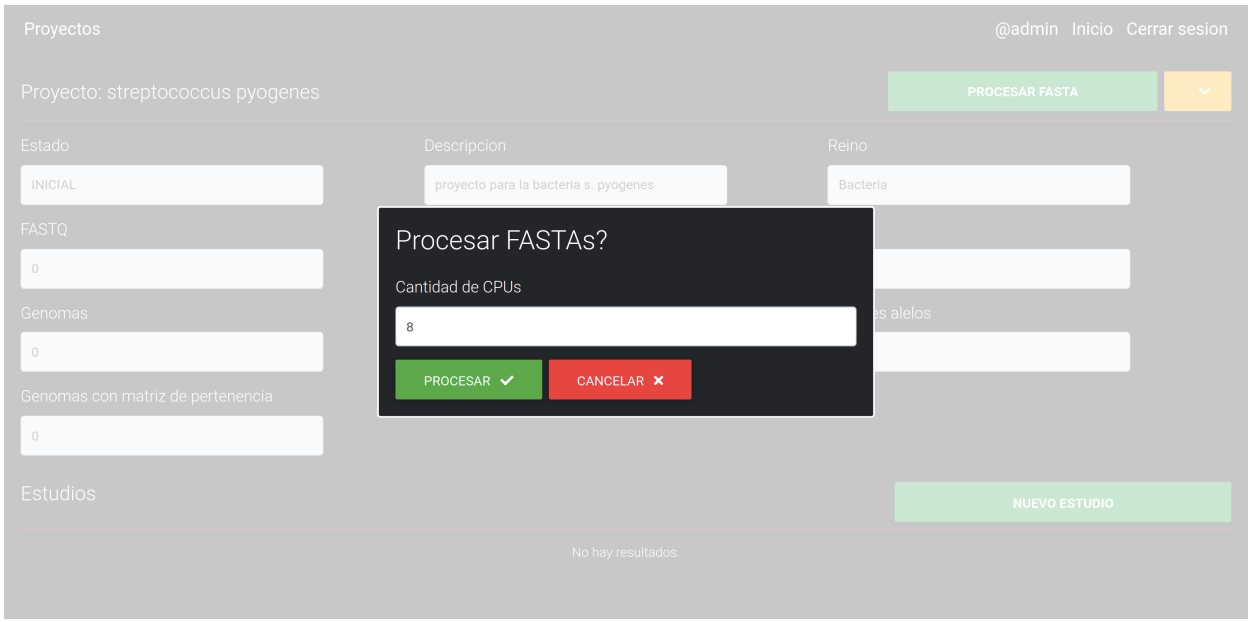


Figura 7.14: Detalle de proyecto - Procesar FASTA

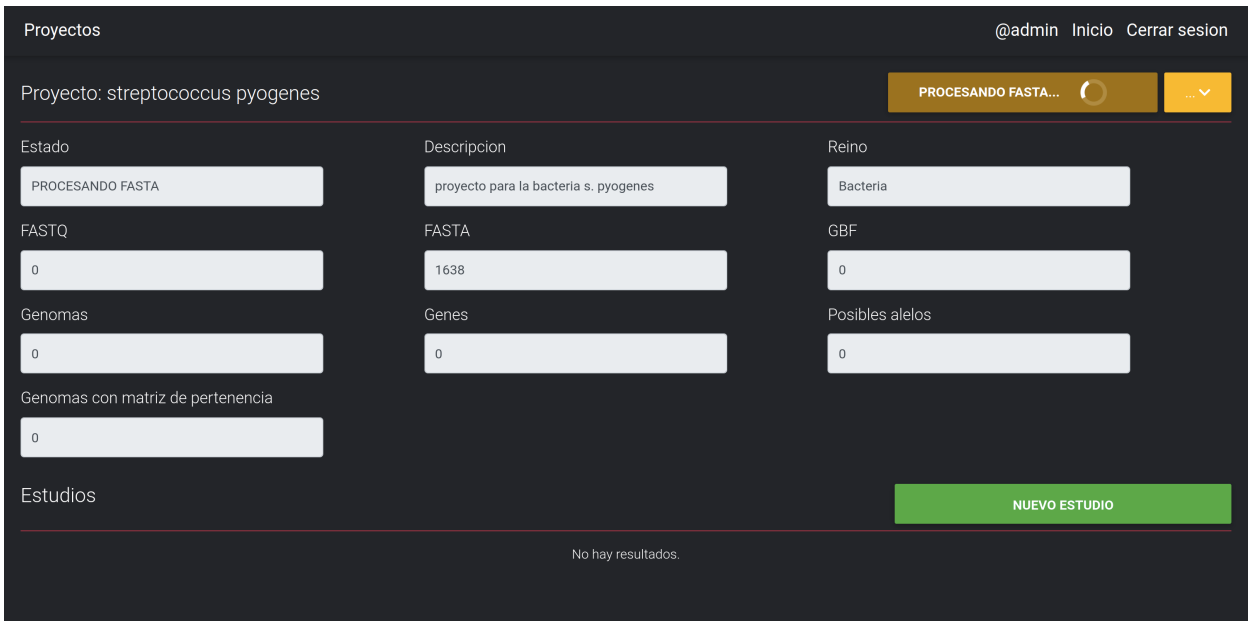


Figura 7.15: Detalle de proyecto - Procesando FASTA

Luego de procesar los archivos FASTA y convertirlos a GBF, el estado del proyecto cambia a **Inicial** y se puede ver que el campo de FASTA quedó en 0 y ahora hay 1638 archivos GBF listos para procesar, habilitando la acción **Procesar GBF**.

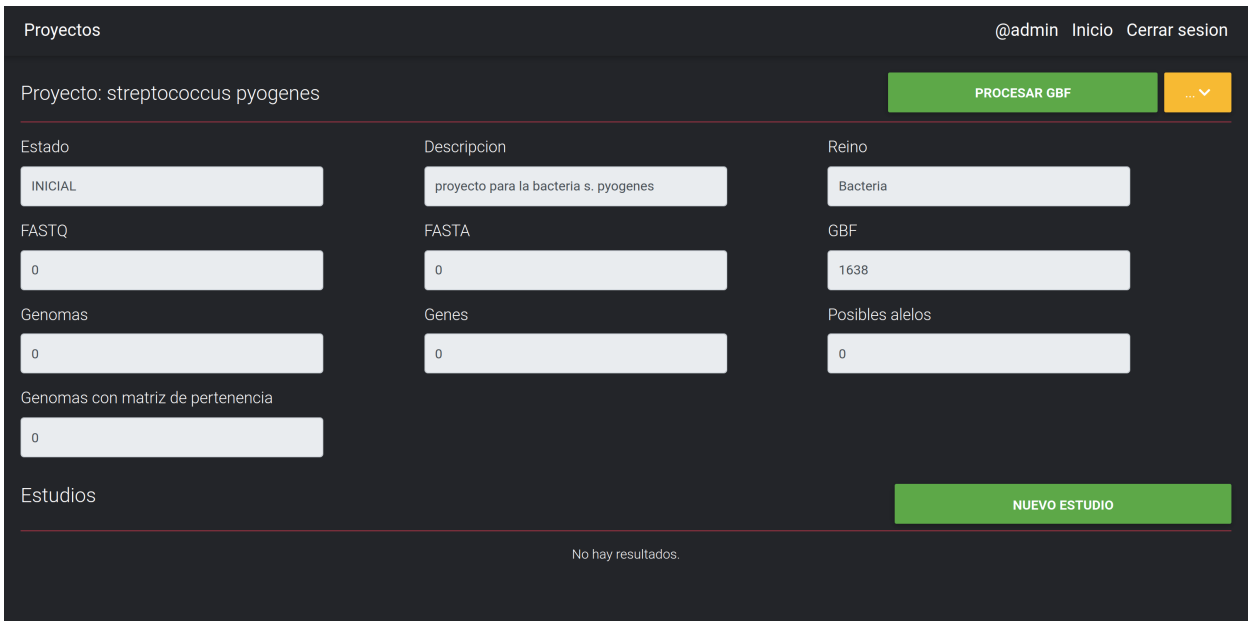


Figura 7.16: Detalle de proyecto - Procesar GBF

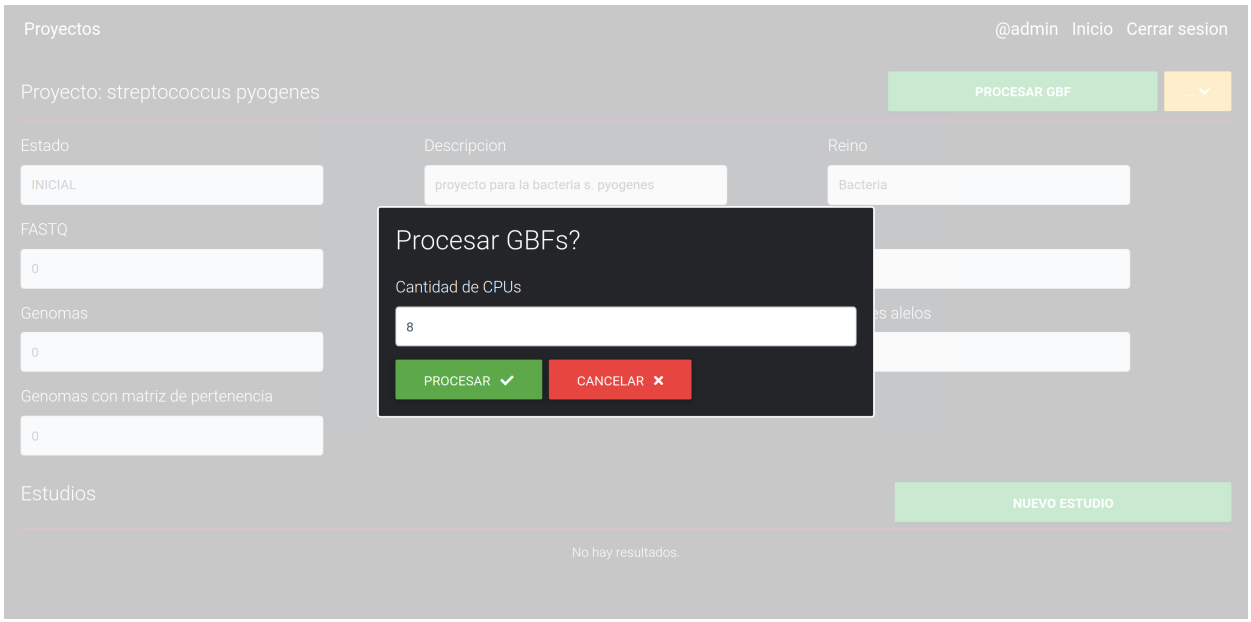


Figura 7.17: Detalle de proyecto - Procesar GBF

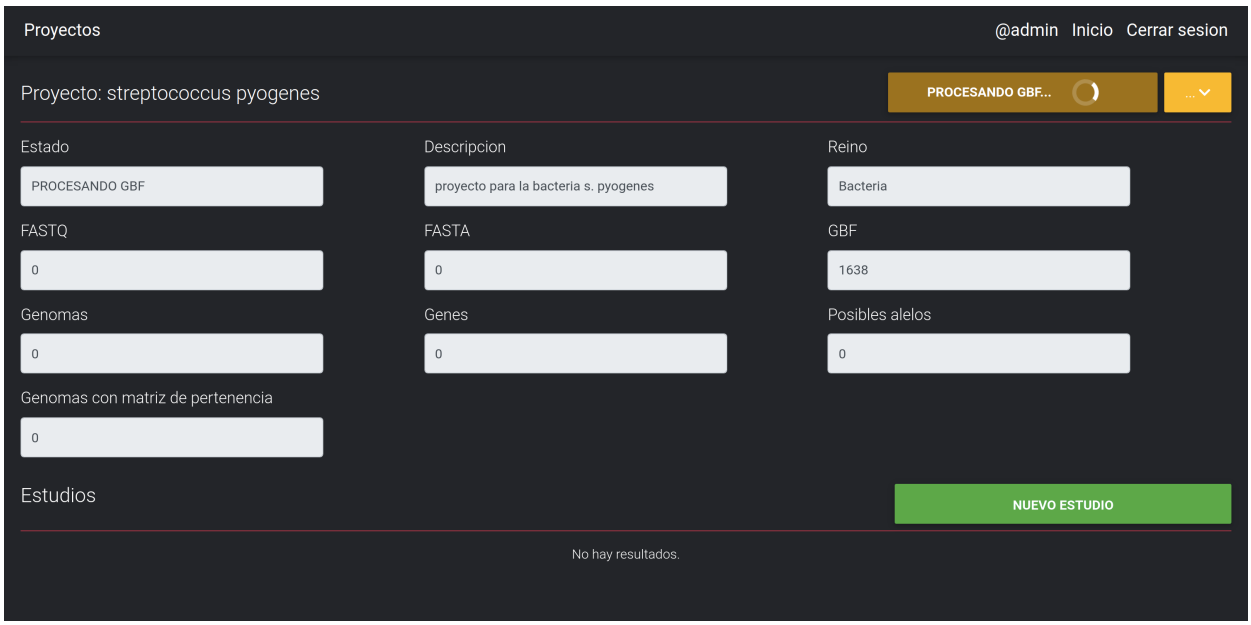


Figura 7.18: Detalle de proyecto - Procesando GBF

En esta instancia se realizan los pasos mencionados para el procesamiento de GBFs, eso implica:

- Insertar genomas en el índice de genomas del proyecto.
- Insertar los genes en el índice de genes del proyecto.
- Insertar alelos en el índice de alelos del proyecto.
- Generar matrices de pertenencia de genes y de alelos para cada genoma.
- Eliminar archivos GBF.
- Cambiar el estado del proyecto a **Listo para estudios**.

En la siguiente imagen se puede ver como queda el detalle del proyecto:

- 1638 genomas insertados
- 158.579 genes
- 82.520 alelos
- 1638 genomas con matrices de pertenencia

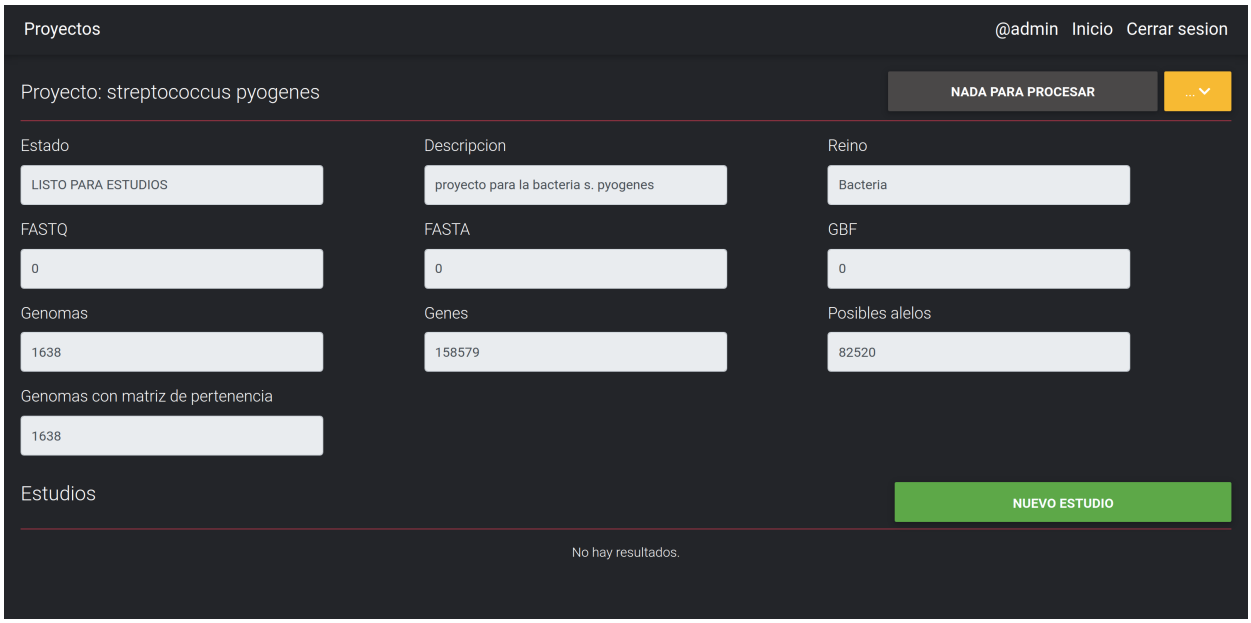


Figura 7.19: Detalle de proyecto - Listo para estudios

Listado de estudios

Como se aprecia en la imagen anterior, debajo del detalle del proyecto se muestra una tabla para listar los estudios del existentes y un botón para crear un estudio.

Creación de estudio

Para crear un estudio se puede abrir el formulario de creación mediante el botón **Nuevo estudio**. En el formulario hay tres campos a completar con nombre, descripción y tipo de estudio deseados. El nombre debe ser único entre los estudios existentes, la descripción es opcional y el tipo de estudio determina si el estudio debe usar genes o alelos. En las tres imágenes que continuan se ve el formulario durante la creación y luego la respuesta exitosa del estudio creado.

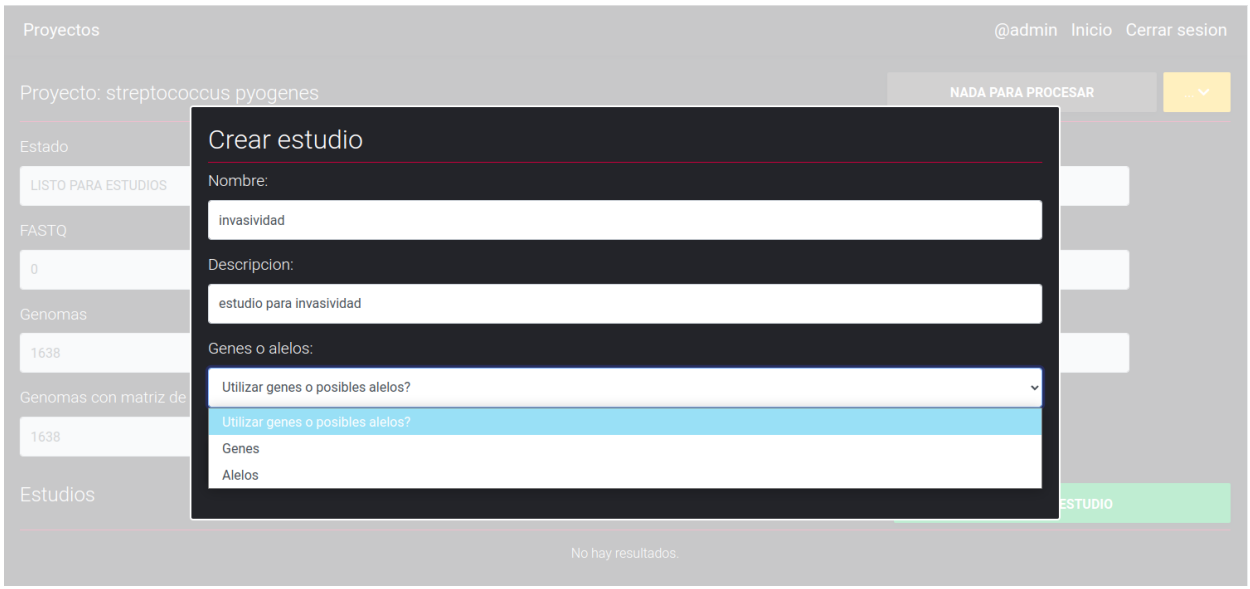


Figura 7.20: Creación de estudio - Completando formulario

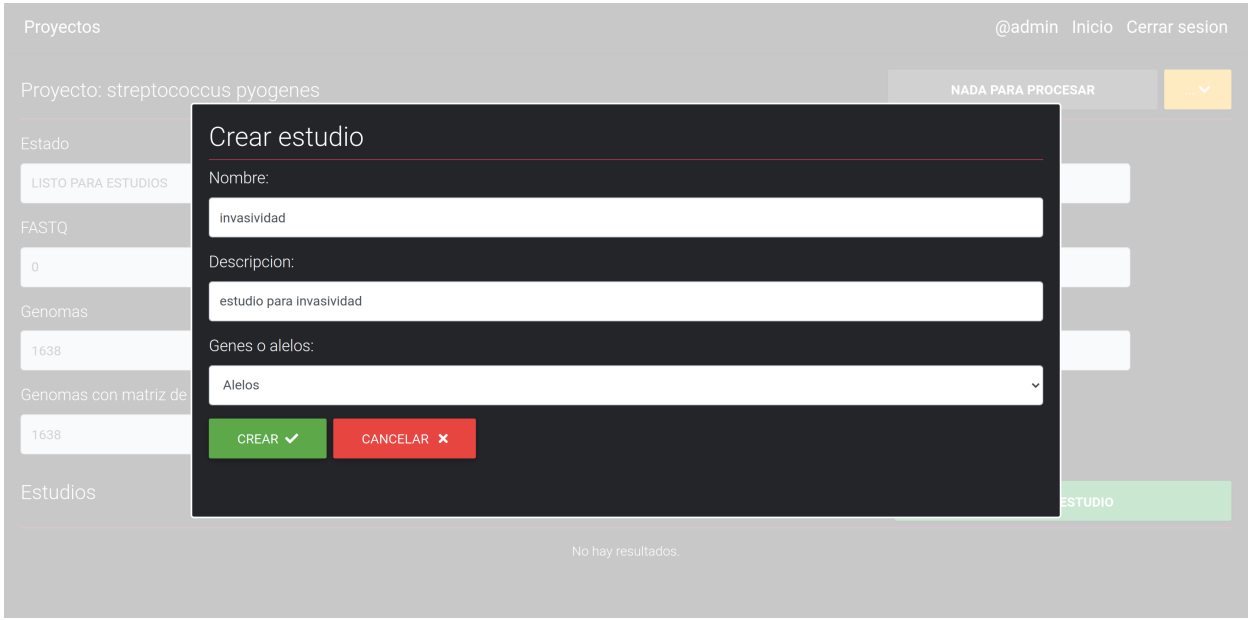


Figura 7.21: Creación de estudio - Formulario completo

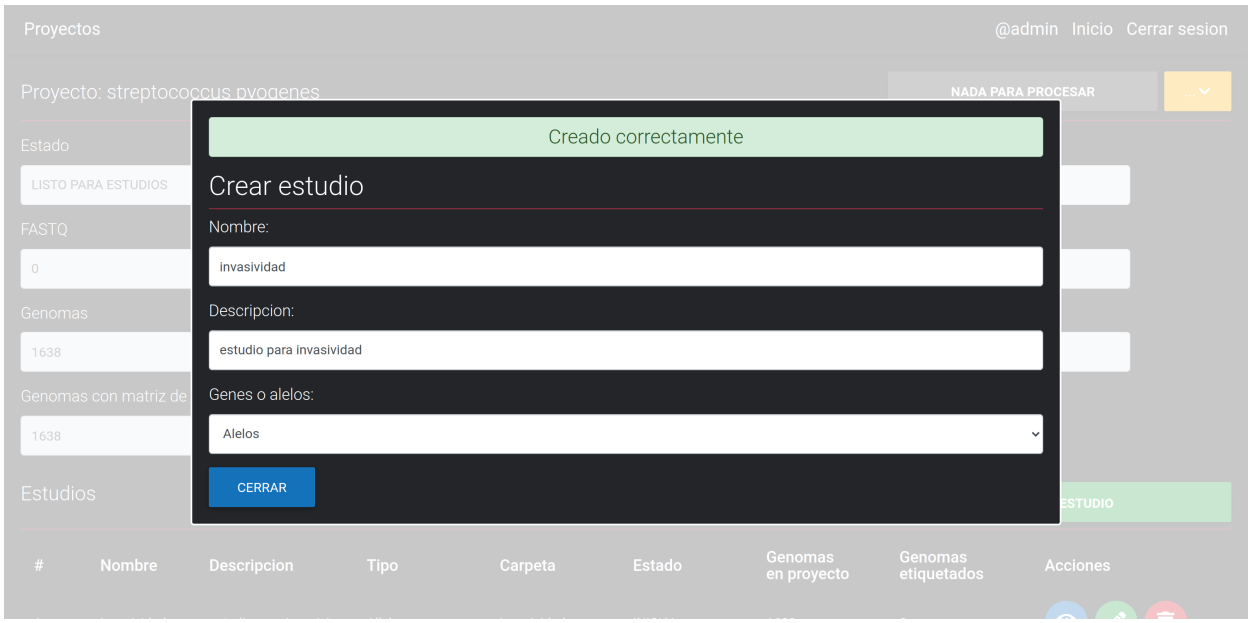


Figura 7.22: Creación de estudio - Estudio creado

Luego de crear un estudio, el mismo aparecerá en el listado de estudio, visualizando su nombre, descripción, el tipo de estudio, la carpeta creada para el mismo, el estado del estudio, la cantidad de genomas que tiene el proyecto, la cantidad de genomas etiquetados para dicho estudio, y algunas acciones para editar, eliminar o acceder al estudio.

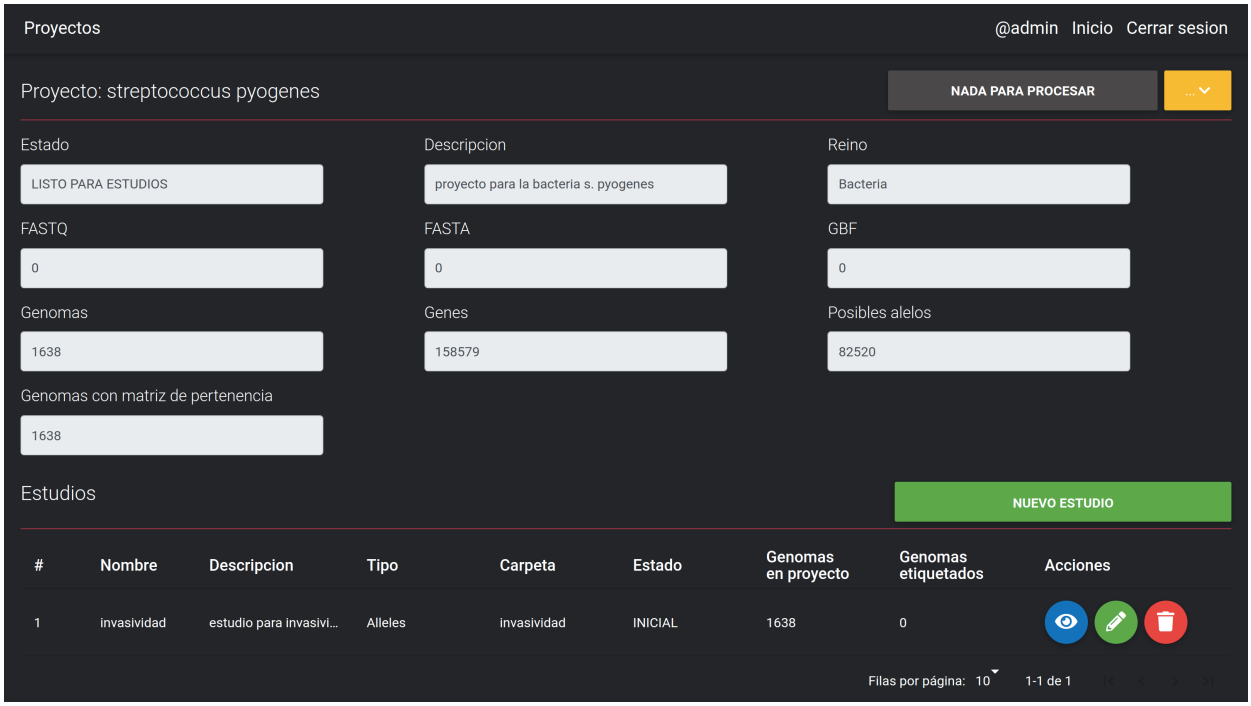


Figura 7.23: Listado de estudios

Edición de estudio

Para un estudio existente se puede editar la descripción abriendo el formulario de edición desde la tabla de estudios. En las dos imágenes a continuación se muestra la edición de un estudio.

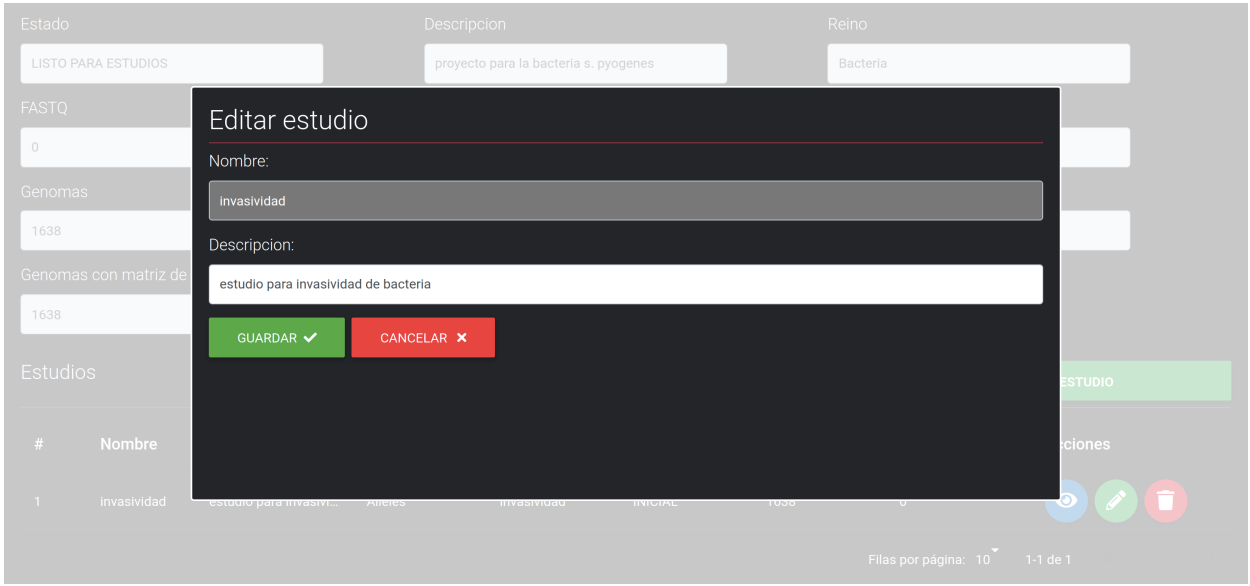


Figura 7.24: Edición de estudio - Editando descripción

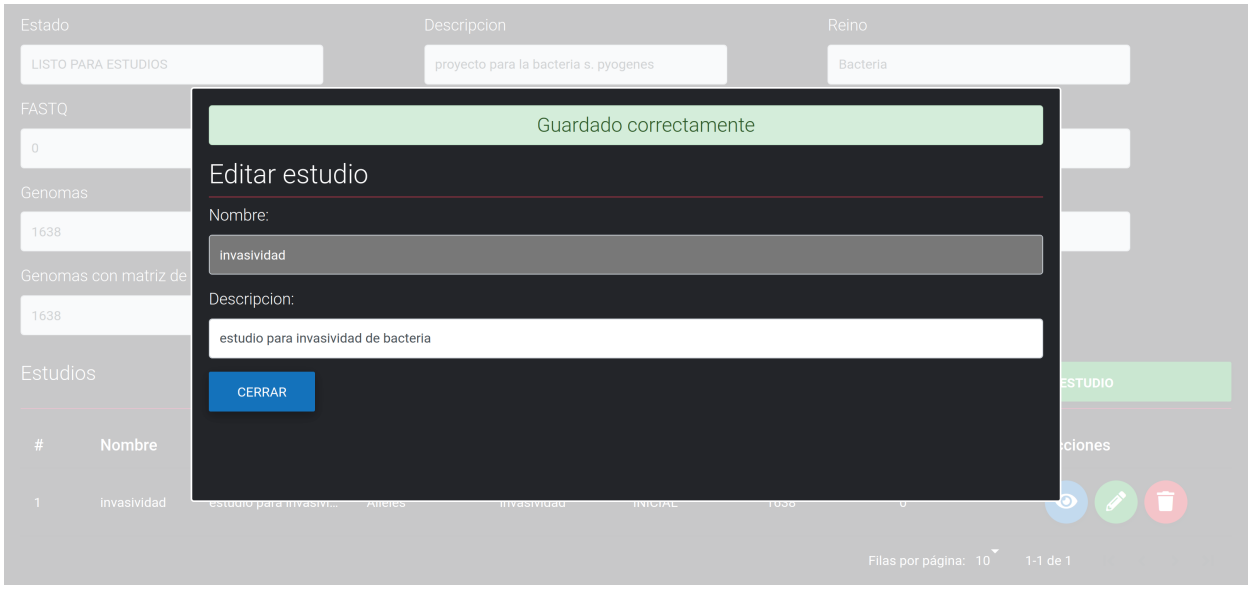


Figura 7.25: Edición de estudio - Estudio editado

Eliminación de estudio

Un estudio existente se puede eliminar, esto implica eliminar las etiquetas de los genomas en el proyecto, el estudio en ElasticSearch y las carpetas creadas para el mismo (modelo creado, logs, archivos, etc.). Es una operación que no puede deshacerse. En la siguiente imagen se ve la eliminación de un estudio.

Figura 7.26: Eliminación de estudio

7.1.3. Detalle de estudio

El último módulo del flujo de trabajo es el del detalle de un estudio, al cual se puede acceder mediante un botón disponible para cada estudio existente en el listado de estudios de un proyecto. En este módulo se ve toda la información de un estudio, y es donde se pueden realizar todas las acciones disponibles para el mismo.

En la imagen siguiente se visualiza un ejemplo del detalle de un estudio. En la parte superior se muestra el nombre y las acciones que se pueden realizar según el estado del estudio y del proyecto, y como vemos en este caso, las acciones iniciales son **Cargar etiquetas** y **Exportar CSV**. Debajo hay campos de texto que muestran el estado, la descripción, la carpeta del estudio, la cantidad de genomas en el proyecto, la cantidad de genomas etiquetados para el estudio, la cantidad de etiquetados con SI y NO, el tipo

de estudio, la arquitectura definida para el modelo del estudio y la cantidad de archivos de genomas detectados para procesar y evaluar por el modelo.

Figura 7.27: Detalle de proyecto - Inicial

Etiquetas

Lo primero que se debe definir para un estudio son sus etiquetas, esto quiere decir etiquetar los genomas correspondientes para luego utilizarlos en el entrenamiento, lograr un modelo con una tasa de acierto aceptable y luego poder realizar evaluaciones o analizar el modelo entrenado y obtener los genes de mayor impacto para investigarlos.

Para cargar las etiquetas podemos exportar un CSV a completar con el valor de cada etiqueta para cada genoma. Esta acción se puede llevar a cabo con el botón **Exportar CSV**, la cual genera un listado con todos los genomas del proyecto que se pueden etiquetar.

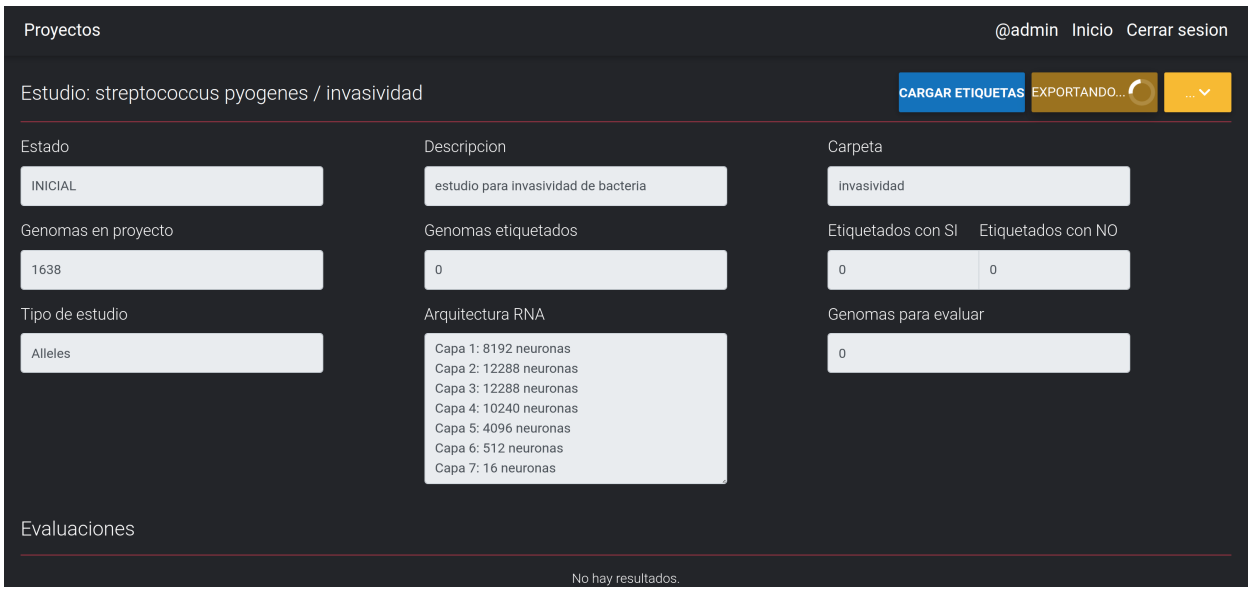


Figura 7.28: Detalle de estudio - Exportando CSV

En la imagen siguiente se muestra el CSV generado, el cual cuenta con dos columnas: una para el nombre de la muestra y la otra para completar con el valor de la etiqueta (SI o NO) para dicha muestra.

	A	B	C	D	E	F	G	H	I
1	Genoma	Etiqueta							
2	ERR026930								
3	ERR026934								
4	ERR026937								
5	ERR026938								
6	ERR026939								
7	ERR026941								
8	ERR028579								
9	ERR028580								
10	ERR028581								
11	ERR028583								
12	ERR028584								
13	ERR028587								
14	ERR028588								
15	ERR033188								
16	ERR033189								
17	ERR033190								

Figura 7.29: Detalle de estudio - CSV para completar con etiquetas

Una vez que se dispone del CSV con las etiquetas y las muestras deseadas, las mismas se pueden cargar a la herramienta con el botón **Cargar etiquetas**.

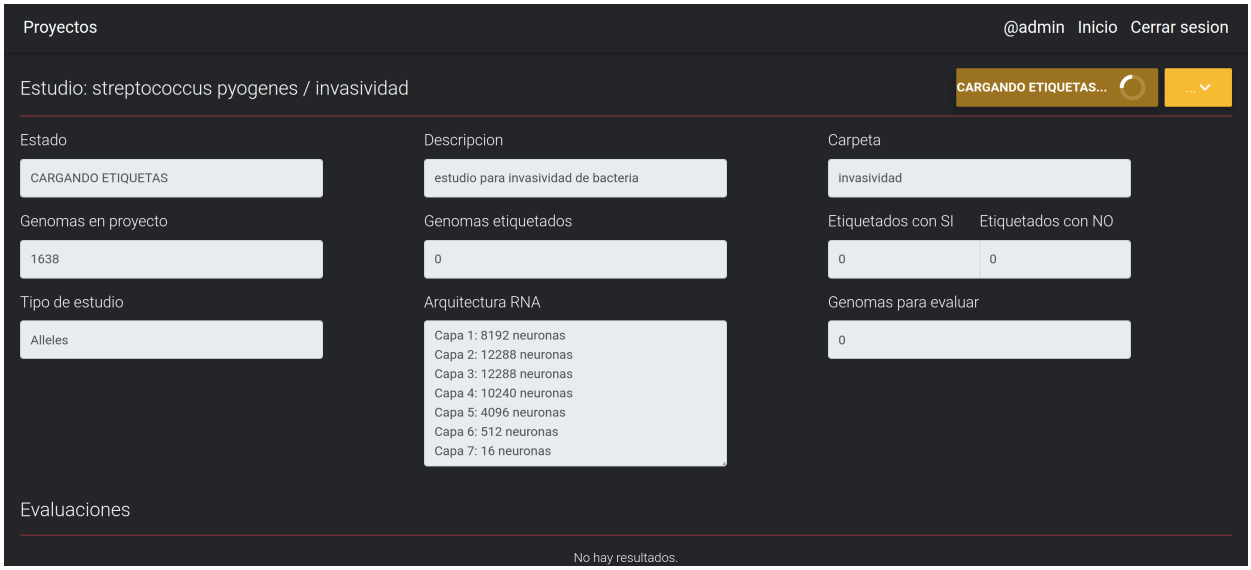


Figura 7.30: Detalle de estudio - Cargando etiquetas

Al finalizar la carga de las etiquetas, en el detalle del estudio se ven actualizadas la cantidad de genomas etiquetados y la cantidad etiquetados con SI y con NO. Además el estado del estudio pasa a **Listo para entrenar**, lo que permite crear el modelo del estudio definiendo la arquitectura del modelo y los parámetros deseados para utilizar en su entrenamiento.

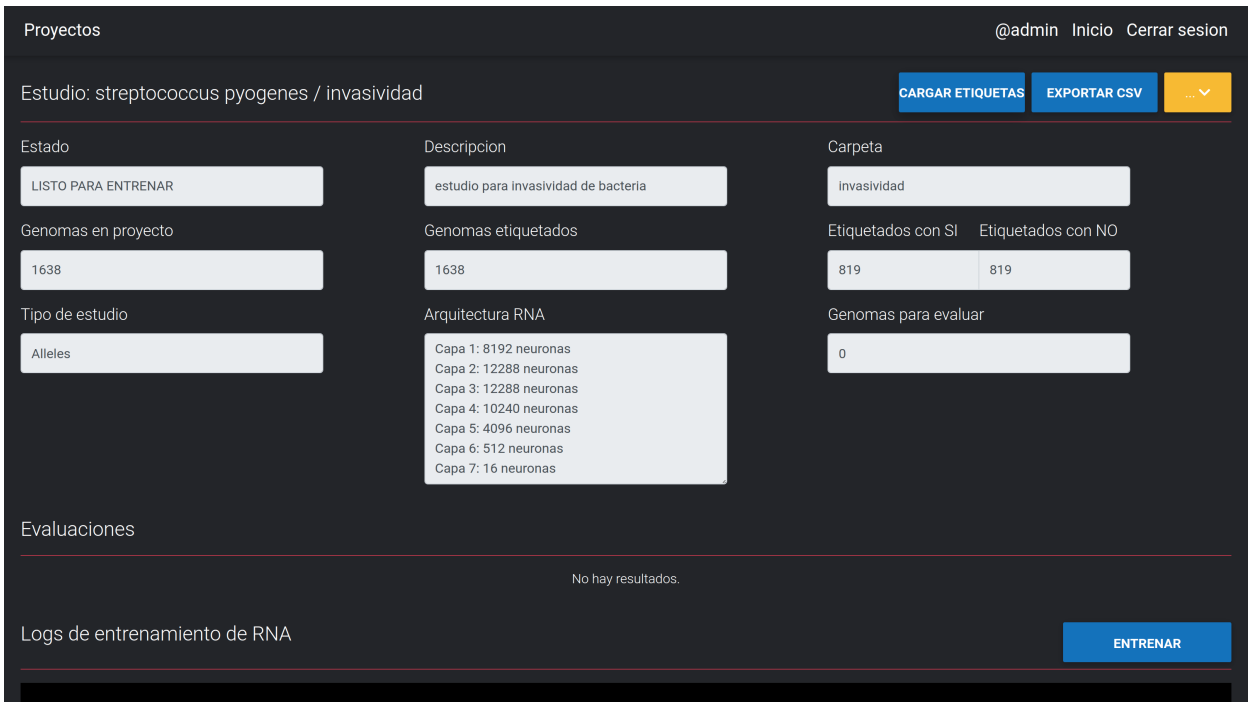


Figura 7.31: Detalle de estudio - Listo para entrenar

Entrenamiento y optimización

Para entrenar el modelo, una vez que el estado del estudio es **Listo para entrenar**, en la sección **Logs de entrenamiento de RNA** se dispone de un botón **Entrenar**, el cual abre un formulario con los campos necesarios para especificar cómo debe ser la arquitectura y el entrenamiento de la red. Como se ve en la imagen que sigue, los parámetros posibles son: la cantidad de pasadas, el porcentaje de tolerancia de diferencia entre conjuntos de etiquetas, el porcentaje de datos a usar en entrenamiento y testeo, el learning rate y la arquitectura de la red.

Figura 7.32: Detalle de estudio - Entrenar

Cuando empieza el entrenamiento, el estudio cambia su estado a **Entrenando**, y no permite otras acciones hasta que termine.

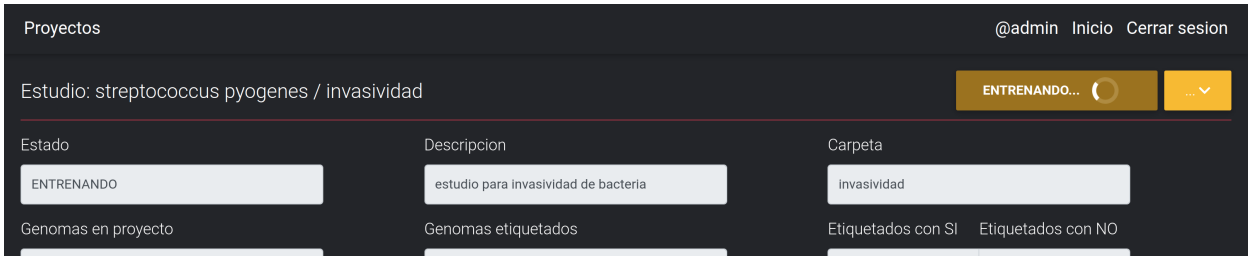


Figura 7.33: Detalle de estudio - Entrenando

En la sección **Logs de entrenamiento de RNA** se puede ir viendo como evoluciona el aprendizaje del modelo. En las primeras líneas del log se detallan algunos números según los genomas existentes y los parámetros ingresados:

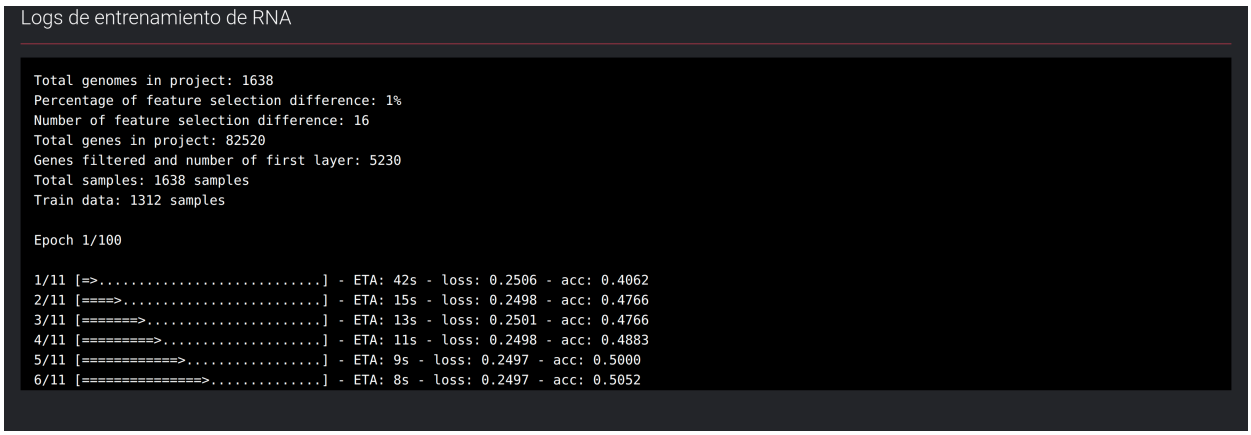


Figura 7.34: Detalle de estudio - Logs de entrenamiento

Al finalizar el entrenamiento, el estado del estudio cambia a **Listo para evaluar**, y en los logs de entrenamiento se pueden apreciar los resultados del último entrenamiento realizado, observando la tasa de acierto y el error en la parte de testeo, que en este caso luego de hacer las primeras 100 pasadas se obtuvo una tasa de acierto del 76 % y un error del 17 % para los datos de testeo.

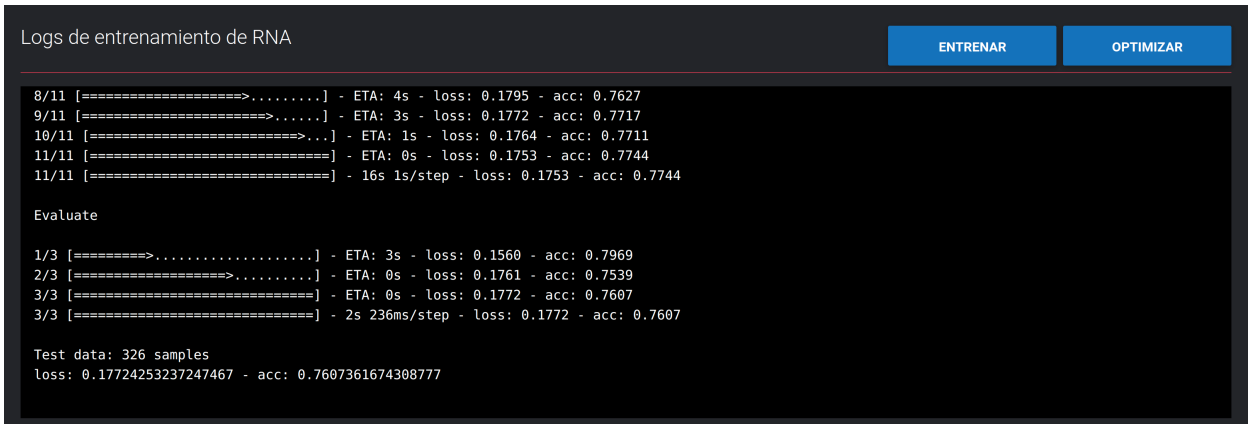


Figura 7.35: Detalle de estudio - Logs de entrenamiento

Como muestra la imagen anterior, al finalizar un entrenamiento, se habilitan las acciones **Entrenar** y **Optimizar**. Entrenar implica crear un modelo de cero, especificando todos los parámetros ya mencionados previamente, y optimizar nos permite ejecutar más pasadas sobre un modelo ya entrenado

para que, con los mismos parámetros ya definidos en el entrenamiento, siga aumentando su tasa de acierto y reduciendo el error.

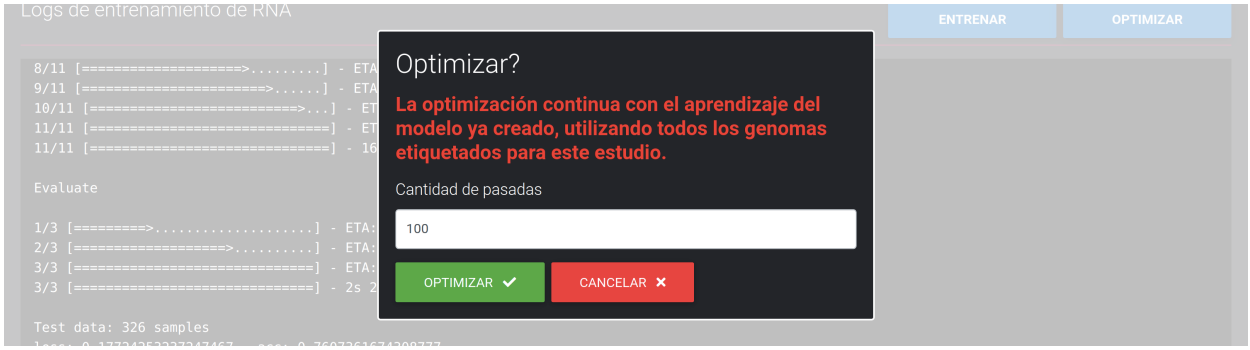


Figura 7.36: Detalle de estudio - Optimizar

Al empezar la optimización, el estudio cambia su estado a **Entrenando**, y no permite otras acciones hasta que termine. En la sección **Logs de entrenamiento de RNA** se puede ir viendo como continua creciendo el aprendizaje del modelo, y como la tasa de acierto parte con el valor finalizado en el entrenamiento anterior.

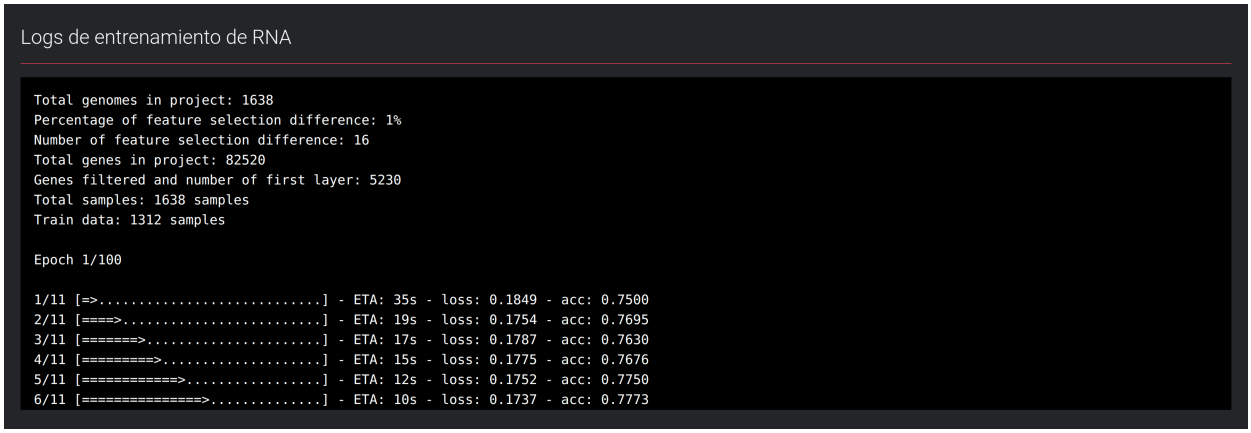


Figura 7.37: Detalle de estudio - Logs de entrenamiento

Al finalizar la optimización, el estado del estudio cambia a **Listo para evaluar**, y en los logs de entrenamiento se pueden apreciar los resultados, una tasa de acierto del 78 % y un error del 17 % para los datos de testeo.

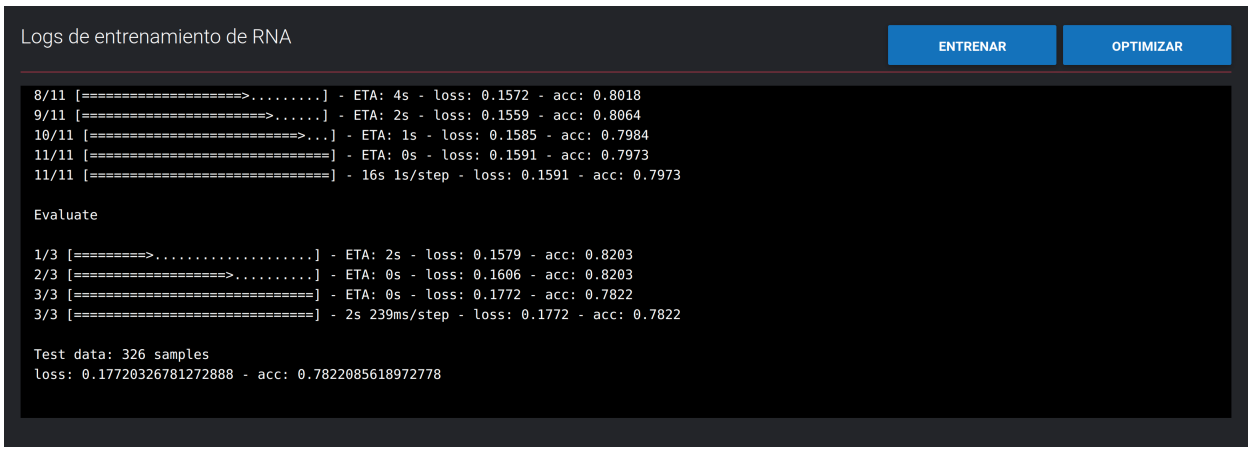


Figura 7.38: Detalle de estudio - Logs de entrenamiento

Evaluación

Para hacer una evaluación sobre un genoma con el modelo entrenado, se debe ingresar el genoma en formato FASTQ, FASTA o GBF en la carpeta **genomes_to_evaluate** del estudio. La herramienta detecta cuando hay algún archivo válido y en el campo **Genomas para evaluar** muestra la cantidad detectada. Si hay por lo menos un archivo y el estado del estudio es **Listo para evaluar** se habilita la acción **Evaluar** a la derecha de la sección **Evaluaciones**.

Figura 7.39: Detalle de estudio - Evaluar

Al hacer click en el botón **Evaluar** se abre una confirmación donde se puede seleccionar la cantidad de CPUs deseadas para realizar la evaluación. Este proceso requiere convertir un archivo FASTQ a FASTA, FASTA a GBF, luego extraer la información del GBF, y con esa información estructurada hacer una predicción mediante el modelo. Cuando empieza la evaluación, el estudio cambia de estado a **Evaluando**.

Figura 7.40: Detalle de estudio - Evaluar

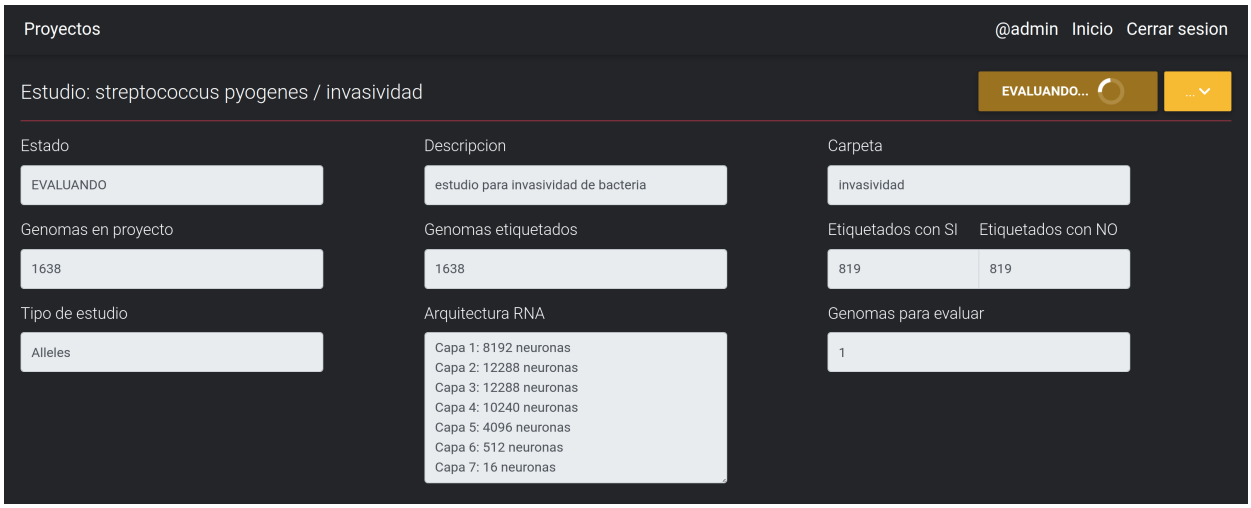


Figura 7.41: Detalle de estudio - Evaluando

Terminada la evaluación, el estudio vuelve a **Listo para evaluar**, el campo **Genomas para evaluar** vuelve a cero, y en la sección **Evaluaciones** se agrega una fila donde se pueden descargar los resultados de la evaluación.

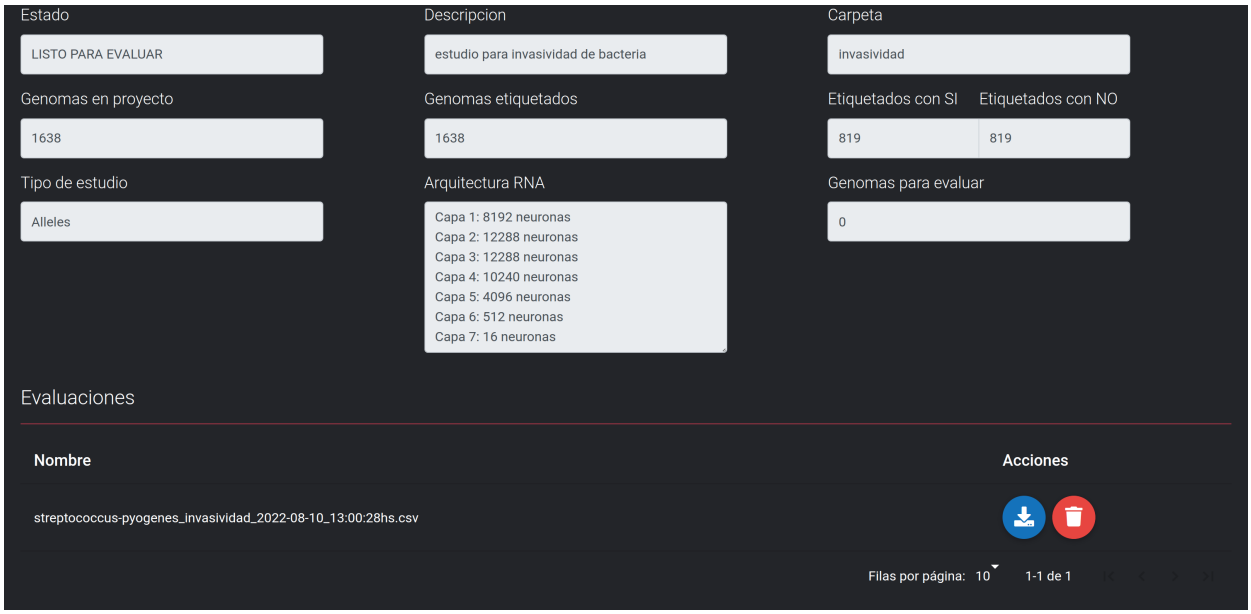


Figura 7.42: Detalle de estudio - Evaluación

Los resultados muestran en un CSV cual fue la predicción hecha para cada genoma evaluado, indicando además con qué porcentaje de acierto determina ese resultado.

	A	B	C	D	E	F	G	H
1	Genome	Prediction	Accuracy					
2	SRR5853557	SI	0.7396038					
3								
4								
5								

Figura 7.43: Detalle de estudio - Resultado evaluación

Análisis del modelo

La última sección que tiene un estudio es el análisis del modelo correspondiente. Para ver los genes con mayor impacto en el resultado se puede analizar el modelo con el botón como se muestra en la imagen siguiente. Mientras se realiza el análisis, el estudio cambia su estado a **Analizando modelo**.

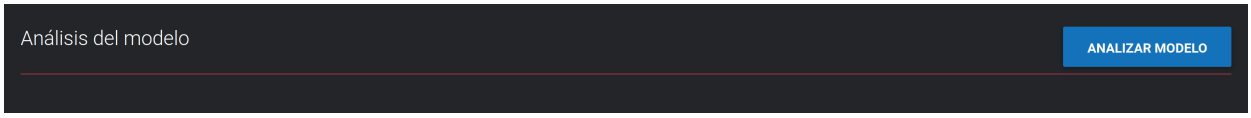


Figura 7.44: Detalle de estudio - Analizar modelo

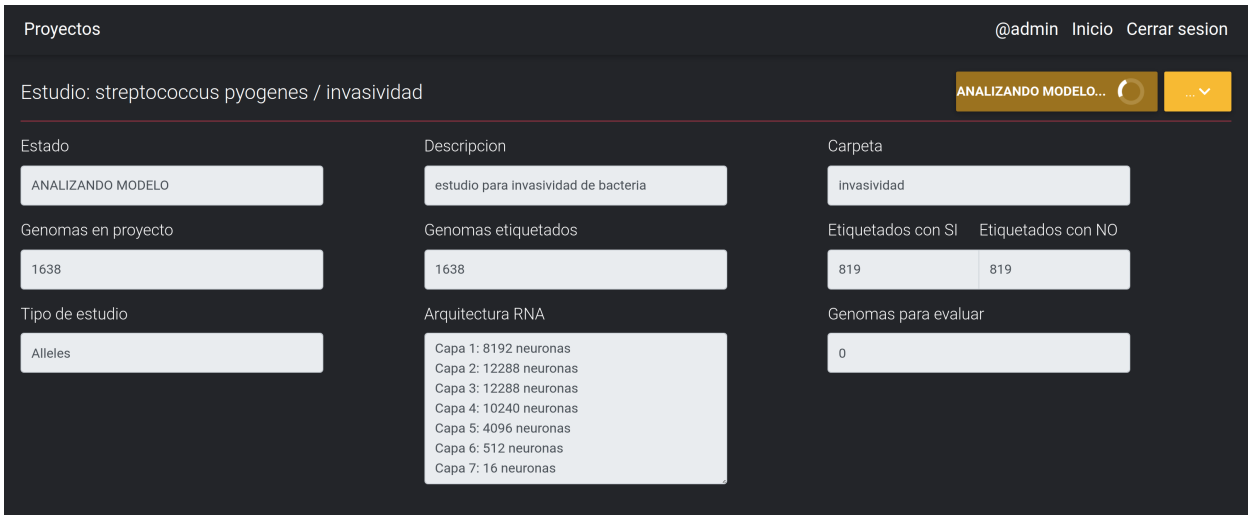


Figura 7.45: Detalle de estudio - Analizando modelo

Al finalizar el análisis, se pueden apreciar dos grandes columnas, a la izquierda una con un gráfico que contiene los 50 genes con mayor impacto en el resultado y cuál es el impacto. En la columna de la derecha se listan esos 50 genes, mostrando para cada uno su id, el nombre del gen (en caso de no ser hypothetical protein), su product y algunas acciones (**Multifasta**, **Alinear multifasta**, **Generar IQtree** y **Detalle**).

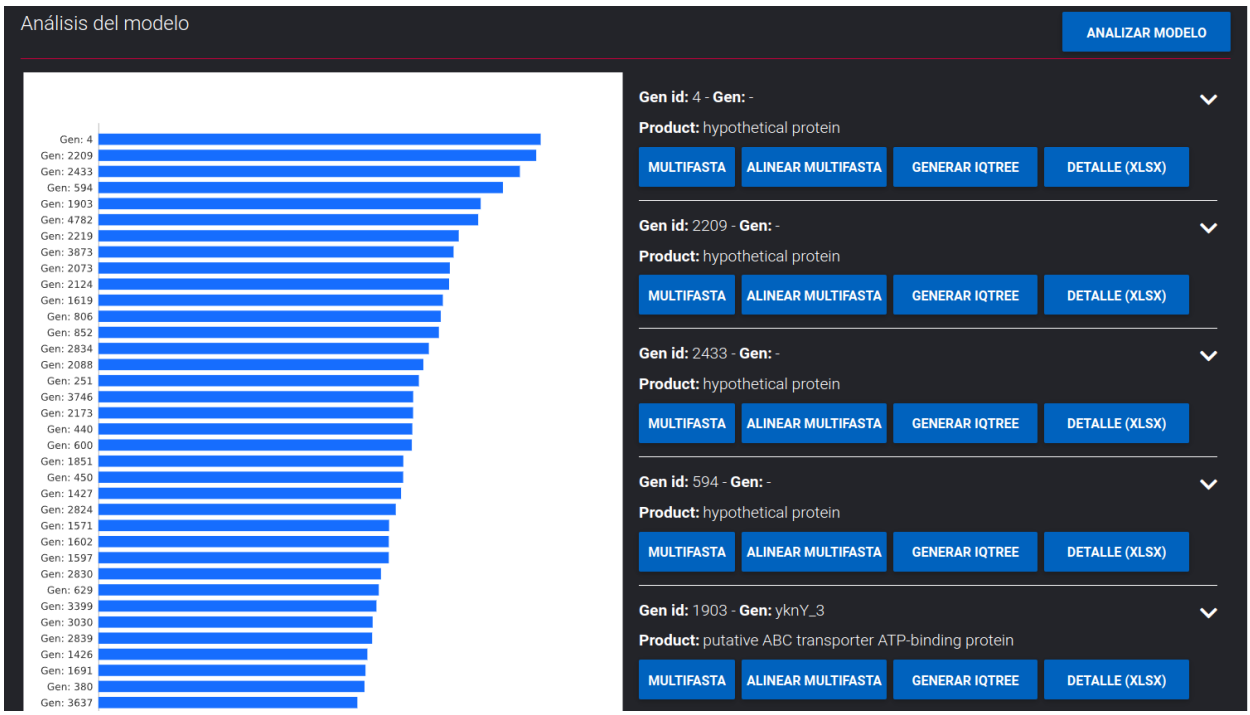


Figura 7.46: Detalle de estudio - Análisis de modelo

Si se despliega la flecha para ver más información se observarán además todos los translations encontrados para ese gen (en caso de no ser hypothetical protein), la cantidad de genomas en los que está presente, cuántos de ellos están etiquetados con SI y cuántos con NO.

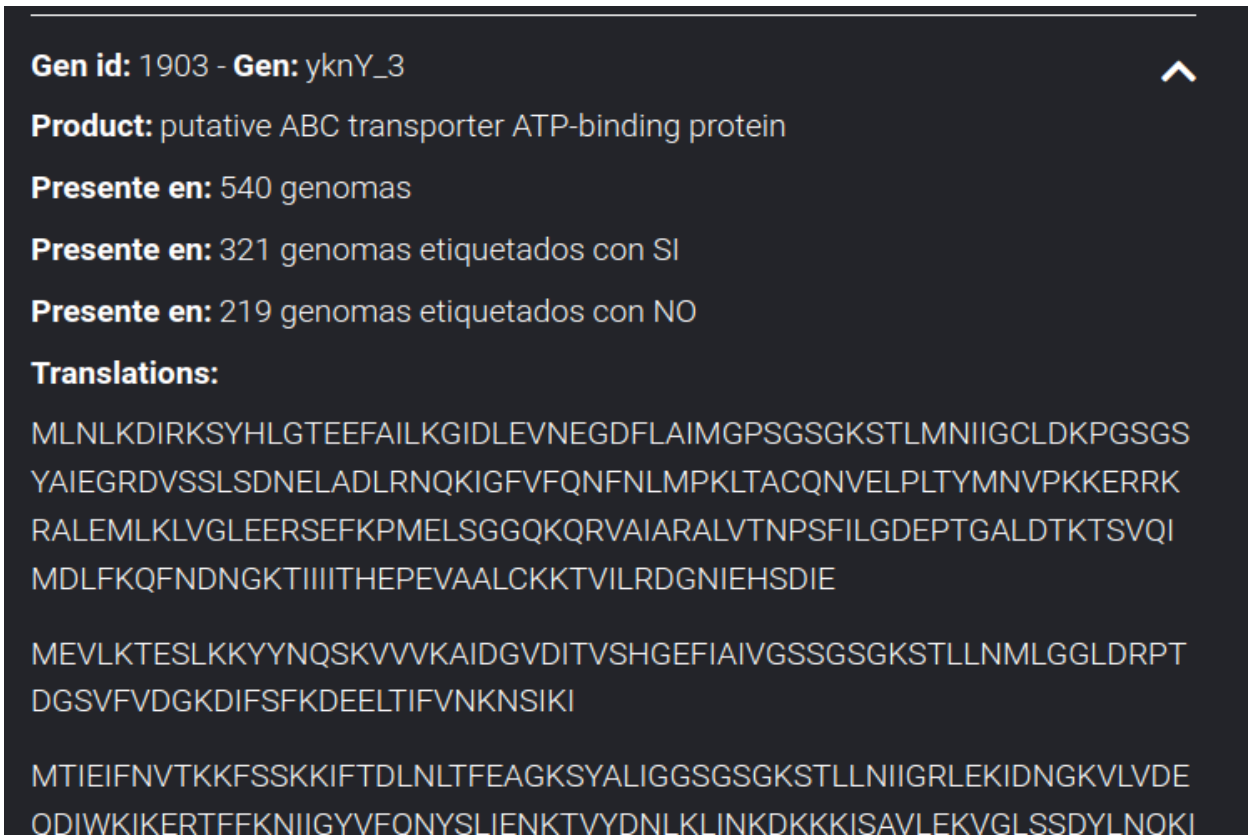


Figura 7.47: Detalle de estudio - Información de gen

Cada gen posee cuatro acciones: **Multifasta**, **Alinear multifasta**, **Generar IQtree** y **Detalle**. El botón **Multifasta** genera instantáneamente el multifasta para el gen. El botón **Alinear multifasta** cambia el estado del estudio a **Alineando multifasta** mientras se realiza dicha acción, al finalizar se habilita el botón **Descargar alineamiento**. En las siguientes imágenes se puede observar:

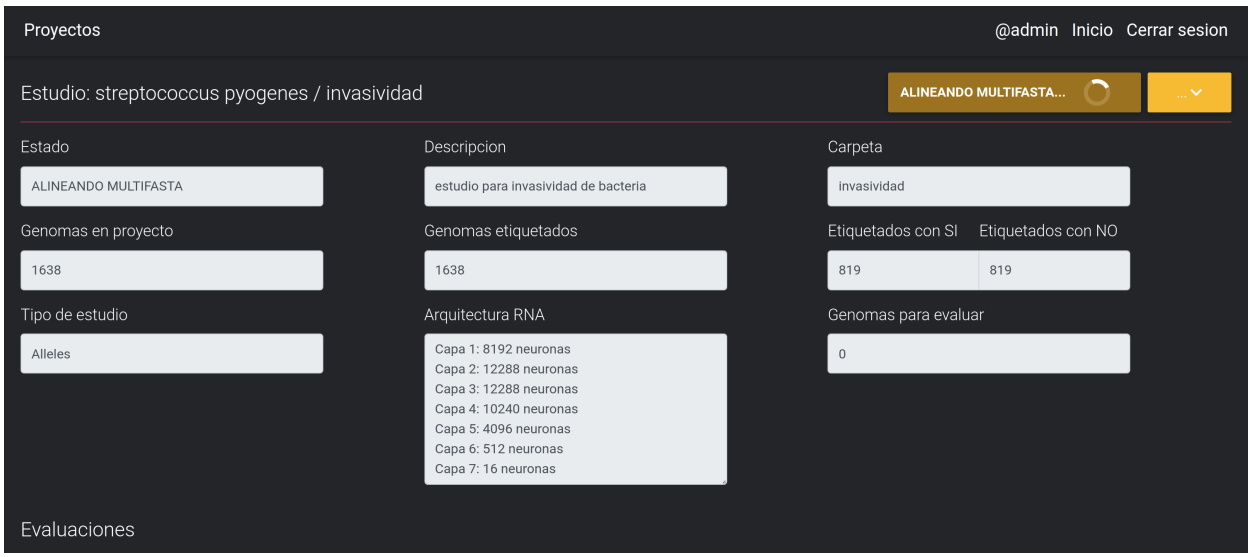


Figura 7.48: Detalle de estudio - Alineando multifasta

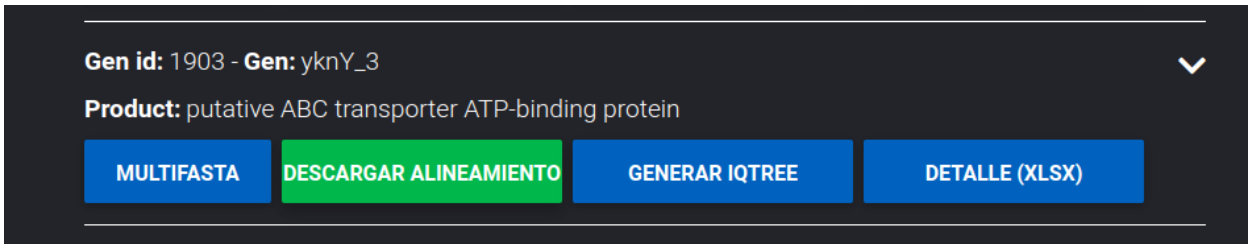


Figura 7.49: Detalle de estudio - Descargar alineamiento

El botón **Generar IQtree** cambia el estado del estudio a **Generando IQtree** mientras se realiza dicha acción, al finalizar se habilita el botón **Descargar IQtree**, como se muestra en las imágenes a continuación:

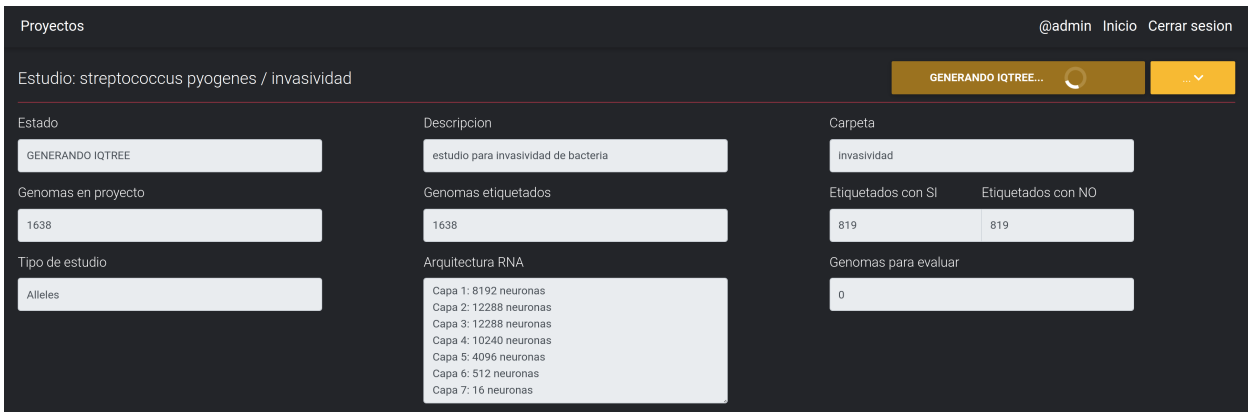


Figura 7.50: Detalle de estudio - Generando IQtree

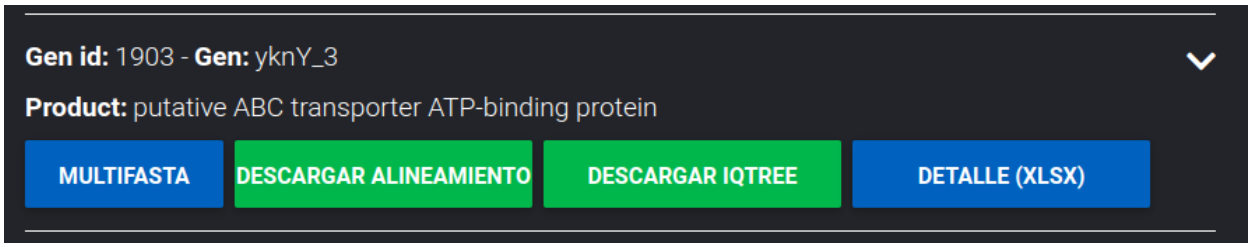


Figura 7.51: Detalle de estudio - Descargar IQtree

Por último, el botón **Detalle** genera el detalle para un gen:

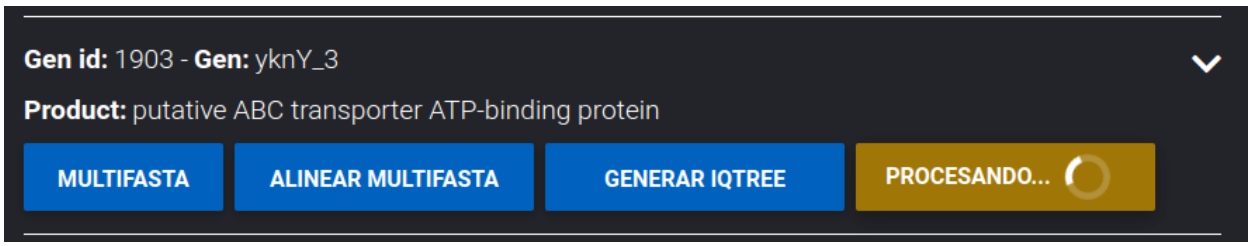


Figura 7.52: Detalle de estudio - Detalle gen

	A	B	C	D
1	Translation	Presente en	Etiquetados con si	Etiquetados con no
2	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYA	1457	702	755
3	MTIEIFNVTKKFSSKKIFTDLNLTFEAGKSYALIGGSGSGKSTLLNIIGRLEKIDNGKVLVDEQI	410	231	179
4	MSHEKKQLMQLSNIVKSYQNGDQVLKVLKGINLTVYEGEFLAIMGPSGSGKSTLMNIIGLLD	1020	492	528
5	MKLINKDKKKISAVLEKVGLSSDYLNQKIYELSGGQAQRVAIARMLMKPRKIILADEPTGALD	3	2	1
6	MEVLKTESLKKYYNQSKVVVKAIDGVDITVSHGEFIAIVGSSGSGKSTLLNMLGGLDRPTDG	2	2	0
7	MKLINKDKKISAVLEKVGLSSDYLNQKIYELSGGQAQRVAIARMLMKPRKIILADEPTGALDG	1	1	0
8	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYA	12	10	2
9	MEVLKTESLKKYYNQSKVVVKAIDGVDITVSHGEFIAIVGSSGSGKSTLLNMLGGLDRSTDG	1	1	0
10	MTIEIFNVTKKFSSKEIFTDLNLTFEAGKSYALIGGSGSGKSTLLNIIGRLEKIDKGKVLVDEQI	1	0	1
11	MPDKGKKHPKQLSGGQQQRVAIARAIVNRPSFIIADEPTGALDSKTSEEILTLFQQLNNEGV	2	2	0
12	MTIEIFNVTKKFSSKKIFKDLNLTFEAGKSYALIGGSGSGKSTLLNIIGRLEKIDNGKVLVDEQ	3	1	2
13	MIGGSGSGKSTLLNIIGRLEKIDNGKVLVDEQDIWKIKERTFFKNIIGYVFQNYSLIENKTVYD	3	0	3
14	MTIEIFNVTKKFSSKEIFTDLNLTFEAGKSYALIGGSGSGKSTLLNIIGRLEKIDNGKVLVDEQI	1	1	0
15	MTIEIFNVTKKFSSKEIFTDLNLTFEAGKSYALIGGSGSGKSTLLNIIGRLEKIDKGKVLVDEQI	1	0	1
16	MSHEKKQLMQLSNIVKSYQNGDQVLKVLKGINLTVYEGEFLAIMGPSGSGKSTLMNIIGLLD	47	35	12
17	MTIEIFNVTKKFSSKEIFTDLNLTFEAGKSYALIGGSGSGKSTLLNIIGRLEKIDNGKVLVDEQI	27	4	23
18	MTIEIFNVTKKFSSKKIFTDLNLTFEAGKSYALIGGSGSGKSTLLNIIGRLEKIDNGKVLVDEQI	2	2	0
19	MSHEKKQLMQLSNIVKSYQNGDQVLKVLKGINLTVYEGEFLAIMGPSGSGKSTLMNIIGLLD	6	6	0
20	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYA	7	7	0
21	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYA	4	4	0
22	MSHEKKQLMQLSNIVKSYQNGDQVLKVLKGINLTVYEGEFLAIMGPSGSGKSTLMNIIGLLD	10	10	0
23	MSHEKKQLMQLSNIVKSYQNGDQVLKVLKGINLTVYEGEFLAIMGPSGSGKSTLMNIIGLLD	104	40	64
24	MSHEKKQLMQLSNIVKSYQNGDQVLKVLKGINLTVYEGEFLAIMGPSGSGKSTLMNIIGLLD	63	31	32
25	MEVLKTESLKKYYNQSKVVVKAIDGVDITVSHGEFIAIVGSSGSGKSTLLNMLGGLDRPTDG	106	99	7
26	MFVNRKIGFVFQSYNLVPVLNVYENIVLPIELDGNDVDEEYVDSIITALGLKEKINSMPNQLSG	104	99	5
27	MTIEIFNVTKKFSSKKIFTDLNLTFEAGKSYALIGGSGSGKSTLLNIIGRLEKIDNGKVLVDEQI	2	0	2
28	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYA	22	10	12
29	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYA	3	1	2
30	MTIEIFNVTKKFSSKEIFTDLNLTFEAGKSYALIGGSGSGKSTLLNIIGRLEKIDNGKALVDEQI	1	1	0

Figura 7.53: Detalle de estudio - Detalle gen hoja 1

	A	B	C
1	Genoma	Etiqueta	Translation
2	ERR227094	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
3	ERR227095	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
4	ERR228429	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
5	ERR228431	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
6	ERR228433	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
7	ERR228434	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
8	ERR228436	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
9	ERR228444	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
10	ERR228447	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
11	ERR228449	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL
12	ERR228450	1	MLNLKDIRKSYHLGTEEFAILKGIDLEVNEGDFLAIMGPSGSGKSTLMNIIGCLDKPGSGSYAIEGRDVSSLSDNELADLRNQKIGFVFQNFNLMPKLTACQNVELPL

Figura 7.54: Detalle de estudio - Detalle gen hoja 2

Capítulo 8

Conclusiones y trabajo futuro

Se logró realizar un software para investigadores que ayuda en el análisis de propiedades biológicas sobre miles de secuencias de genomas completos de un organismo mediante técnicas de Machine Learning, el cual permite realizar predicciones y encontrar los genes de mayor impacto, que en caso de ser genes no clasificados hasta la fecha, resultan de interés para su posterior análisis en laboratorio.

La herramienta reduce la brecha entre las dos ciencias, al permitir analizar grandes volúmenes de información mediante una interfaz amigable a biólogos con poca experiencia en IA.

A su vez, gracias al trabajo en conjunto con investigadores del Centro Nacional de Genómica y Bioinformática (CNGB), se extendió la funcionalidad del software para agilizar el trabajo del día a día de los científicos, como son la generación de archivos multifasta, el alineamiento de los mismos y la generación de árboles IQTREE sobre los genes de mayor impacto en un estudio.

Se pudo además encontrar soluciones informáticas a problemas técnicos reales, partiendo del entendimiento del dominio de la biología con ayuda de especialistas en el tema, para poder procesar información cruda de gran tamaño y complejidad y así estructurarla teniendo presente la resolución del objetivo de la tesina.

Se consiguió desarrollar un software simple de distribuir, pudiendo ser ejecutado en cualquier servidor que disponga de Docker. Esto brinda portabilidad y pocos pasos para lograr su instalación y uso. El desarrollo fue realizado enteramente en Gitlab, permitiendo el uso y extensión del mismo

por parte de otros usuarios o desarrolladores que así lo deseen.

Se realizó la instalación de la herramienta en el CNGB para el uso por parte de los científicos, con el objetivo de seguir trabajando en conjunto y así mejorar y utilizar el software en nuevos tipos de organismos biológicos.

Trabajo a futuro

- Extender la plataforma para poder indicar una lista de genomas hosteados en la nube y así poder utilizar datos públicos que se encuentren en diferentes portales (ej. NCBI).
- Extender el funcionamiento aprovechando los beneficios de Docker para distribuir el entrenamiento a nivel hardware, para reducir los tiempos de ejecución considerablemente.
- Extender el comportamiento de las redes neuronales que se crean en los estudios para permitir predicciones sobre preguntas no binarias, es decir, más de dos etiquetas posibles para las muestras, dando lugar a la investigación de propiedades biológicas de organismos que requieren una clasificación más compleja.

Bibliografía

[1] A. M. G. S. P. R. G.-R. Santiago Rubio, Rafael Adrián Pacheco-Orozco, "Dna next-generation sequencing (ngs): Present and future in clinical practice," *Universitas Medica*, 2020.

[2] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 ed., 2010.

[4] U. Obolski, A. Gori, J. Lourenço, C. Thompson, R. Thompson, N. French, R. S. Heyderman, and S. Gupta, "Identifying genes associated with invasive disease in s. pneumoniae by applying a machine learning approach to whole genome sequence typing data," *Scientific Reports*, vol. 9, p. 4049, Mar 2019.

[5] R. Dias and A. Torkamani, "Artificial intelligence in clinical and genomic diagnostics," *Genome Medicine*, vol. 11, p. 70, Nov 2019.

[6] D. Ruano-Gallego, J. Sanchez-Garrido, Z. Kozik, E. Núñez-Berrueco, M. Cepeda-Molero, C. Mullineaux-Sanders, J. N.-B. Clark, S. L. Slater, N. Wagner, I. Glegola-Madejska, T. I. Roumeliotis, T. Pupko, L. Ángel Fernández, A. Rodríguez-Patón, J. S. Choudhary, and G. Frankel, "Type iii secretion system effectors form robust and flexible intracellular virulence networks," *Science*, vol. 371, no. 6534, p. eabc9531, 2021.

[7] D. A. Martínez, "Cliente para plataforma de búsqueda de biomarcadores con valor pronóstico/predictivo de cáncer," May 2016.

[8] A. S. Sanyahumbi, S. Colquhoun, R. Wyber, and J. Carapetis, "Global disease burden of group a streptococcus," *National Center for Biotechnology Information*, 2 2016.

[9] J. Carapetis, A. Steer, E. Mulholland, and M. Weber, "The global burden of group a streptococcal diseases," *The Lancet infectious diseases*, vol. 5, pp. 685–94, 12 2005.

[10] R. J. Aumann and L. S. Shapley, *Values of non-atomic games*. Princeton Legacy Library, Princeton, NJ: Princeton University Press, Apr. 2016.

Facultad de Informática
ENSEÑANZA

UNIVERSIDAD NACIONAL DE LA PLATA

Corresponde a Expediente: 3300-003381/20-003

Secretario de Asuntos Académicos:

Cumplo en informar a Ud., que **Ferella, Nicolás** legajo 11687/8, alumno de la carrera **Licenciatura en Informática y Pizio, Pablo Román** legajo 05752/1 alumno de la carrera **Licenciatura en Informática**, por medio de la presente hacen entrega de la Tesina de Licenciatura titulada "IDENTIFICACION DE PROPIEDADES BIOLOGICAS EN ORGANISMOS UTILIZANDO TECNICAS DE MACHINE LEARNING SOBRE SECUENCIAS DE GENOMA COMPLETO", dirigida por la Lic. Claudia Pons.

En relación a la presentación efectuada por los recurrentes, se deja constancia que cumplen con lo determinado en la Resolución 228/21 del Reglamento de Tesina de Grado, aprobada por el Honorable Consejo Directivo.

Dirección de Enseñanza, 27 de agosto de 2022.-

MPM

Prof. JULIETA CASTELLI
Directora
Dirección de Enseñanza
Facultad de Informática - UNLP

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

La Plata, 27 de setiembre de 2022

Sr.Decano
Facultad de Informática
S/D

De mi consideración:

En referencia a lo solicitado por expediente 3300- 2291/20, cumplo en informar que el trabajo de grado “Identificación de propiedades biológicas en organismos utilizando técnicas de Machine Learning sobre secuencias de genoma completo", realizada por Pablo Pizio y Nicolás Ferella y dirigido por Claudia Pons.

- Propuesta presentada en noviembre 2020
- Informe de avance presentado en agosto 2021

La Comisión Evaluadora del trabajo está integrada por los profesores:

- Bazán
- Rucci
- Garrido
- Ronchetti (suplente)

El trabajo no cuenta con respuestas no favorables por parte de los miembros de la Comisión Evaluadora.

Dra. Patricia Bazán
Coordinación de Trabajos de Grado

Exp. 3300-3381/20-003

FACULTAD DE INFORMATICA
COMISION ASESORA DE ENSEÑANZA

La Plata, 18 de octubre de 2022

Esta Comisión aconseja ratificar la Comisión Evaluadora de la tesina de Licenciatura "Identificación de propiedades biológicas en organismos utilizando técnicas de Machine Learning sobre secuencias de genoma completo" de los alumnos Pablo Román PIZIO y Nicolás FERELLA, directora Claudia PONS y asesora profesional Josefina CAMPOS, con los profesores:

Titulares: Patricia BAZÁN, Enzo RUCCI y Alejandra GARRIDO

Suplente: Franco RONCHETTI

C.A.E.

Corresponde Expe. Nº 3300-003381/20-003

La Plata, 20 de Octubre de 2022

VISTO

El informe final de tesina de Licenciatura presentado por Pablo Román Pizio, a través de expediente N° 3300-003381/20-003

la Ord. 307 de la Facultad de Informática de la UNLP

CONSIDERANDO

El dictamen de la Comisión Asesora de Enseñanza del 18/10/2022

----el Honorable Consejo Directivo en su sesión de fecha 20/10/2022, por unanimidad (15 votos)

RESUELVE

ARTICULO 1°.- RATIFICAR la comisión evaluadora de la tesina de Licenciatura titulada: "Identificación de propiedades biológicas en organismos utilizando técnicas de Machine Learning sobre secuencias de genoma completo", presentada por Pablo Román Pizio. Directora: Prof. Claudia Pons. Asesora profesional: Prof. Josefina Campos. Miembros de la comisión evaluadora:

Titulares:

Prof. Patricia Bazán
Prof. Enzo Rucci
Prof. Alejandra Garrido

Suplentes:

Prof. Franco Ronchetti

ARTICULO 2°.- NOTIFÍQUESE de la presente a los interesados. GIRESE a la Dirección de Enseñanza a sus efectos. REGÍSTRESE por Mesa de Entradas y Archivo.

Dr. Marcelo Naiouf
Decano

Mg. Pablo Thomas
Secretario Académico

Resolución HCD Nº 622/22

Facultad de Informática
ENSEÑANZA

UNIVERSIDAD NACIONAL DE LA PLATA

Corresponde a Expediente: 3300-003381/20-003

Visto, habiendo los interesados expuesto la presente Tesina de Licenciatura, según consta en el Acta nro 25832, gírese para la prosecución del trámite.

Dirección de Enseñanza, 2 de diciembre de 2022.
MV

PABLO F. ANGULO
Sub Jefe Dpto.
Dirección Enseñanza