



Universidad Nacional de Entre Ríos

FACULTAD DE INGENIERÍA

TRABAJO INTEGRADOR FINAL

Bases de Datos

Autor:

Justo Garcia

2023

Tabla de contenidos

1	Introducción	2
2	Desarrollo	3
2.1	Descripción de la base de datos	3
2.2	Diseño	3
2.2.1	Diagrama de entidad relación	3
2.2.2	Diccionario de datos	11
2.2.3	Diagrama de tablas	13
2.3	Implementación	14
2.3.1	Declaración	14
2.3.2	Adquisición e inserción de datos	14
2.4	Manipulación	15
2.4.1	Consultas requeridas	15
2.4.2	Uso de interfaz de acceso a datos	21
3	Conclusión	26

1 Introducción

En este informe se desarrollan las actividades planteadas en el marco del trabajo integrador final de bases de datos. En él se partirá de una base de dato de interés bioinformático para repasar los temas dictados en el curso.

2 Desarrollo

En esta sección se describirán los procesos llevados a cabo para la resolución del problema planteado, cualquier detalle de implementación puede ser verificado en el repositorio de GitHub de este trabajo [\[1\]](#)

2.1 Descripción de la base de datos

Para el desarrollo se decidió diseñar, implementar y manipular una base de datos relacional que aloje diversos registros sobre proteínas de unión al ARN y que permita el acceso a esta información.

2.2 Diseño

2.2.1 Diagrama de entidad relación

Para el diseño del diagrama de entidad-relación, opté por analizar los datos disponibles y considerar cómo obtener información adicional a partir de ellos.

Dado el interés en esta familia de proteínas, comencé mi búsqueda de datos útiles para la construcción de la base de datos. Encontré algunas soluciones previamente implementadas, pero estaban desactualizadas. Finalmente, decidí utilizar RBPDB [\[2\]](#), una base de datos que fue mantenida por el Centro de Investigación Celular y Biomolecular de la Universidad de Toronto hasta 2012. Esta base de datos es una de las más completas disponibles y recibió

una excelente recepción en su artículo de presentación. De este recurso pude obtener un archivo csv con diversa información sobre las proteínas identificadas como de unión al ARN de 5 especies. Los datos que consideré relevantes fueron los siguientes:

- UniProtID
- Nombre del gen
- Descripción del gen
- Tax ID
- Nombre de la especie
- Dominios

Partiendo de ella podía extraer los identificadores de UniProt correspondientes a proteínas que habían sido identificadas como pertenecientes a la familia de interés. Esto me sería muy útil ya que UniProt es una de las principales fuentes de información de proteínas y además de proporcionarme datos de interés provee documentación detallada para realizar consultas a la API, permitiendo automatizar el proceso de recolección de datos y expandir en instancias posteriores los datos extraídos de estas consultas.

Para evaluar que información me sería relevante hice pruebas de consulta a la API con Postman [\[3\]](#).

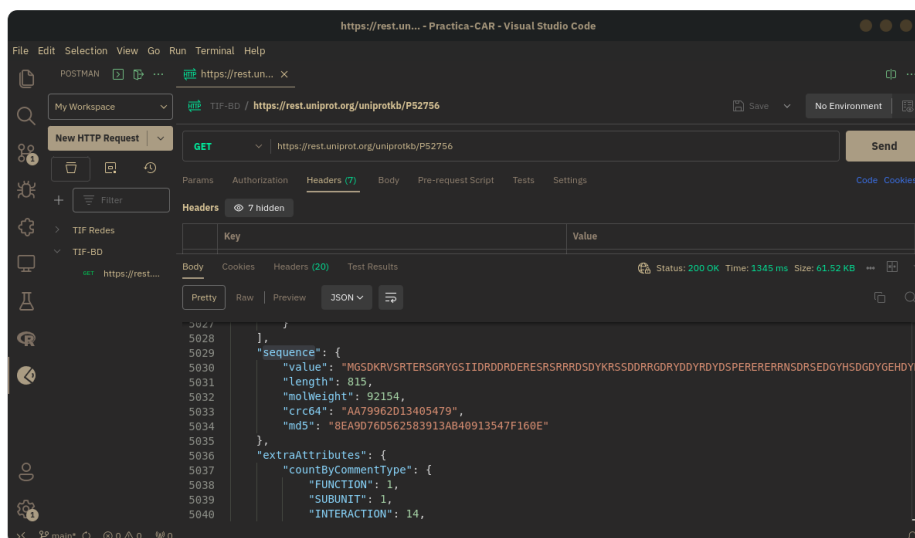
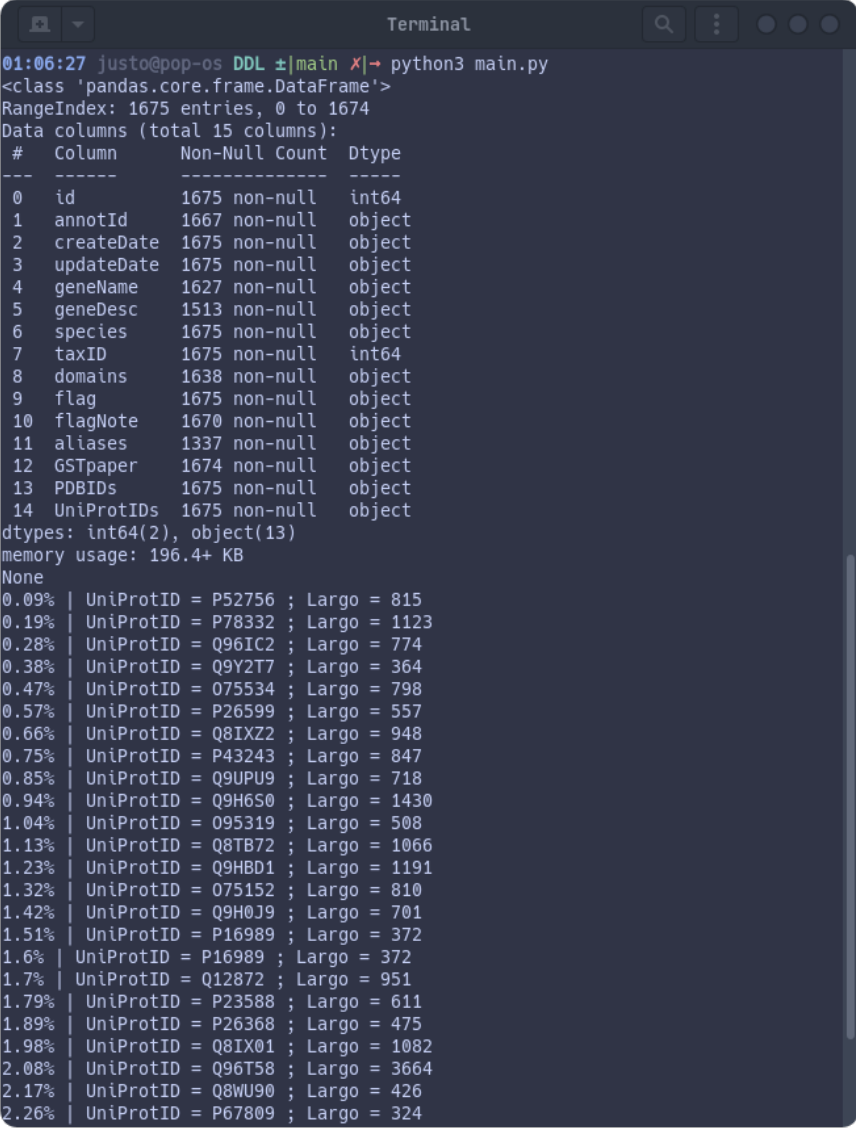


Fig. 1: Software Postman

A partir de analizar la respuesta obtenida en formato JSON, decidí que los datos que iba a extraer de ella eran:

- Secuencia
- Largo de la secuencia
- Referencias
 - Autores
 - Años

Luego realicé los pedidos a la API de forma automática para cada una de las proteínas que tuviese un UniProtID[2].



```
01:06:27 justo@pop-os DDL ±|main X|→ python3 main.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1675 entries, 0 to 1674
Data columns (total 15 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   id           1675 non-null   int64
1   annotId      1667 non-null   object
2   createDate   1675 non-null   object
3   updateDate   1675 non-null   object
4   geneName     1627 non-null   object
5   geneDesc     1513 non-null   object
6   species      1675 non-null   object
7   taxID        1675 non-null   int64
8   domains      1638 non-null   object
9   flag         1675 non-null   object
10  flagNote     1670 non-null   object
11  aliases      1337 non-null   object
12  GSTpaper     1674 non-null   object
13  PDBIDs       1675 non-null   object
14  UniProtIDs   1675 non-null   object
dtypes: int64(2), object(13)
memory usage: 196.4+ KB
None
0.09% | UniProtID = P52756 ; Largo = 815
0.19% | UniProtID = P78332 ; Largo = 1123
0.28% | UniProtID = Q96IC2 ; Largo = 774
0.38% | UniProtID = Q9Y2T7 ; Largo = 364
0.47% | UniProtID = O75534 ; Largo = 798
0.57% | UniProtID = P26599 ; Largo = 557
0.66% | UniProtID = Q8IXZ2 ; Largo = 948
0.75% | UniProtID = P43243 ; Largo = 847
0.85% | UniProtID = Q9UPU9 ; Largo = 718
0.94% | UniProtID = Q9H6S0 ; Largo = 1430
1.04% | UniProtID = O95319 ; Largo = 508
1.13% | UniProtID = Q8TB72 ; Largo = 1066
1.23% | UniProtID = Q9HBD1 ; Largo = 1191
1.32% | UniProtID = O75152 ; Largo = 810
1.42% | UniProtID = Q9H0J9 ; Largo = 701
1.51% | UniProtID = P16989 ; Largo = 372
1.6% | UniProtID = P16989 ; Largo = 372
1.7% | UniProtID = Q12872 ; Largo = 951
1.79% | UniProtID = P23588 ; Largo = 611
1.89% | UniProtID = P26368 ; Largo = 475
1.98% | UniProtID = Q8IX01 ; Largo = 1082
2.08% | UniProtID = Q96T58 ; Largo = 3664
2.17% | UniProtID = Q8WU90 ; Largo = 426
2.26% | UniProtID = P67809 ; Largo = 324
```

Fig. 2: Proceso de obtención de datos

Habiendo obtenido las secuencias correspondientes a cada proteína, decidí utilizar el modulo ProtParam de Biopython [4]. Este esta basado en la famosa herramienta del Expasy [5], otra de las valiosas fuentes de información de proteínas. Analizando las posibilidades que

esta me proveía, decidí que iba a registrar los siguientes datos:

- Peso molecular
- Punto isoelectrico
- Porciones de estructuras secundarias:
 - Fracción de hélice
 - Fracción de giros
 - Fracción de hoja

Toda esta información la fui almacenando en archivos csv, de modo que luego sea fácil su manipulación con la librería pandas [6].

Teniendo en claro cuáles eran los datos con los que iba a contar procedí ahora sí a la identificación de entidades y relaciones entre ellas. De este proceso identifiqué las siguientes entidades:

- **Proteína:** representa las proteínas que pertenecen a la familia de interés. Tendrá atributos específicos como identificadores de bases de datos de relevancia, descripción, dominios, datos estructurales, de secuencia, etc.
- **Secuencia:** la relación con proteína es evidente, pero se la pensó como una entidad aparte por la dificultad de almacenar cadenas de caracteres largas y de alta variabilidad de tamaño sin perder la eficiencia del modelo relacional. Manteniéndolas separadas permitiría mejores tiempos de accesos en los casos que no se requiera de ella.
- **Referencia:** se utiliza para registrar información bibliográfica relacionada con las proteínas de interés. Debe tener atributos como título, año.

- **Autor:** representa a los autores o investigadores que han contribuido a los estudios y publicaciones relacionados.
- **Gen:** se utiliza para registrar información sobre los genes asociados a las proteínas. Cada gen puede estar relacionado con una o varias proteínas y puede contener atributos como el nombre del gen.
- **Especie:** se refiere a las especies a las que pertenecen las proteínas de interés. Cada especie puede tener atributos que describen su nombre común, científico y otros datos de interés.

De estas entidades surgen relaciones entre ellas, las seleccionadas para representar el caso de estudio son las siguientes:

- Proteína $\xrightarrow{\text{Pertenece a}}$ Especie
- Proteína $\xrightarrow{\text{Tiene}}$ Secuencia
- Proteína $\xrightarrow{\text{Codificada por}}$ Gen
- Proteína $\xrightarrow{\text{Referenciada por}}$ Referencia
- Referencia $\xrightarrow{\text{Escrita por}}$ Autor

A partir de haber identificado las entidades y relaciones relevantes para la construcción de la base de datos, procedí a integrarlo en el diagrama de entidad-relación. Para una mejor visualización decidí realizar un esquema con las entidades sin atributos y las relaciones entre ellas [3] y luego esquemas para cada una de las entidades con sus atributos.

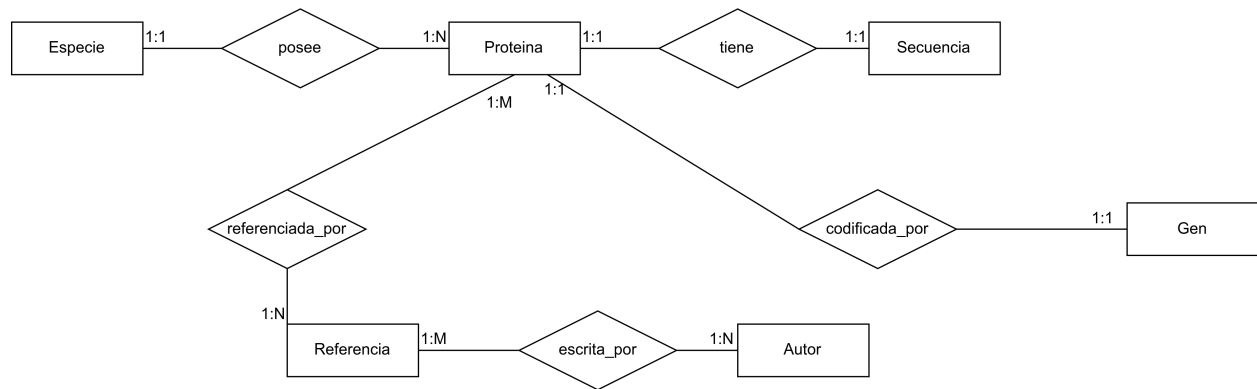


Fig. 3: Diagrama de entidad relación

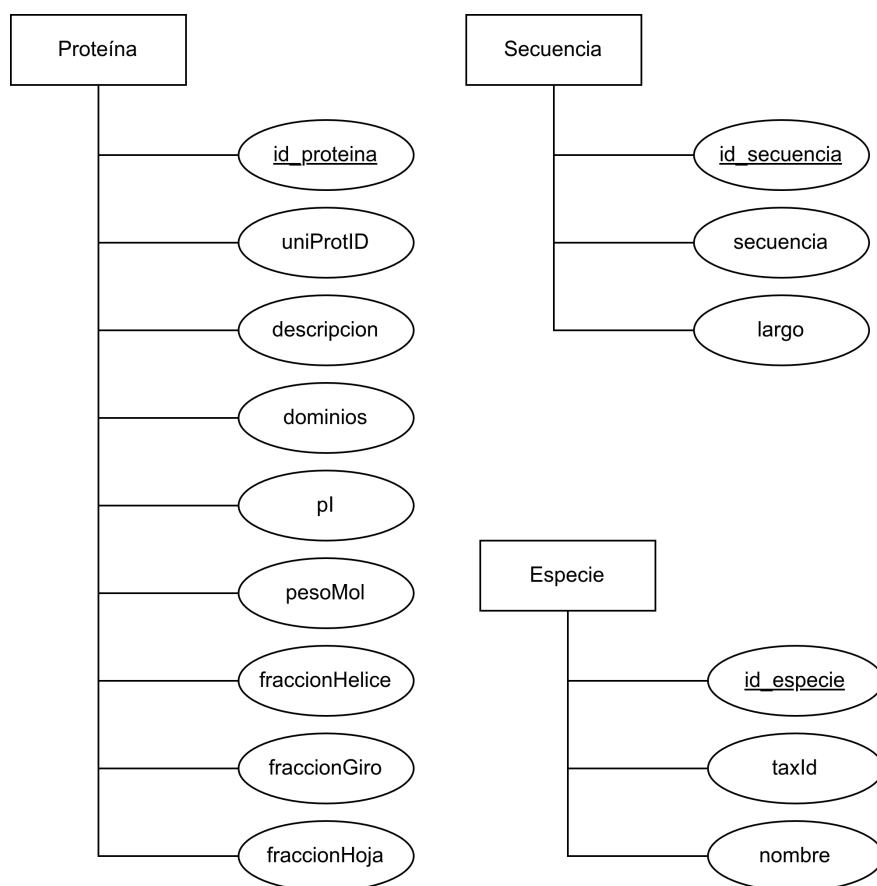


Fig. 4: Proteína, Secuencia, Especie y sus atributos

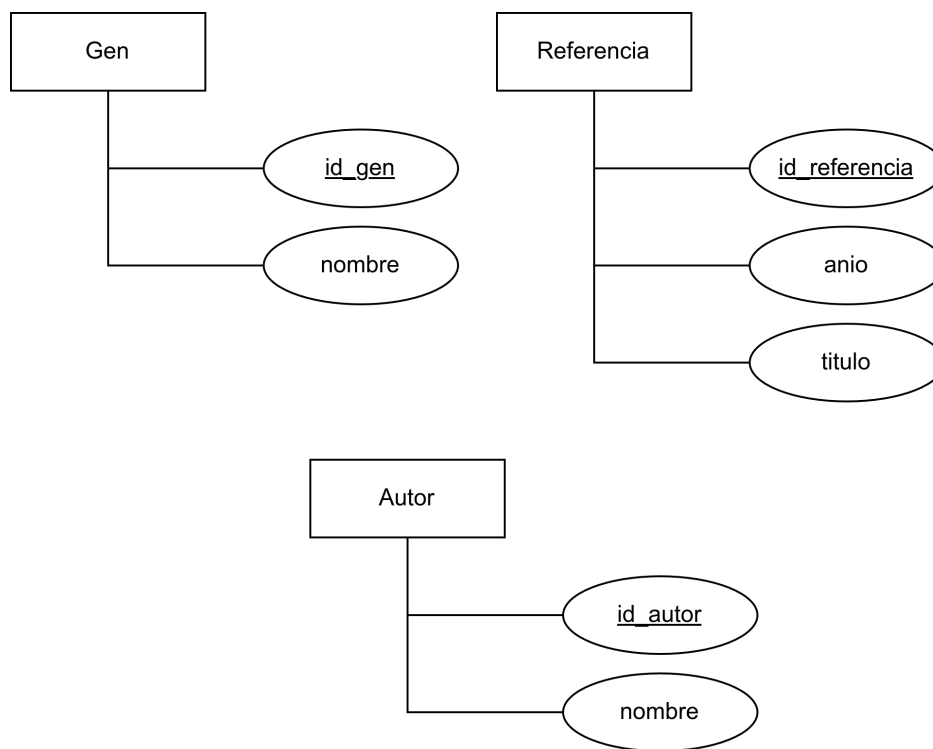


Fig. 5: Gen, Referencia, Autor y sus atributos

2.2.2 Diccionario de datos

De esta forma, puedo conformar el diccionario de datos:

- **Proteína:** Representa una proteína de unión al ARN.
 - id_proteina: Identificador primario interno de la proteína
 - uniProtID: Identificador en UniProt.
 - descripcion: Descripción de la proteína.
 - dominios: Dominios que conforman a la proteína.
 - pI: Punto isoelectrico de la proteína.

- pesoMol: Peso molecular.
- fraccionHelice: Fracción de hélice en su estructura secundaria.
- fraccionGiro: Fracción de giro en su estructura secundaria.
- fraccionHoja: Fracción de hoja en su estructura secundaria.
- id_especie: Foreign key que apunta a la especie a la cual pertenece.

- **Secuencia:**

- id_secuencia: Identificador primario de la secuencia.
- secuencia: Secuencia aminoacídica.
- largo: Largo de la secuencia.
- id_proteina: Foreign key que apunta a la proteína a la cual pertenece.

- **Especie:**

- id_especie: Identificador primario de la especie.
- taxId: Identificador único de categoría taxonómica.
- nombre: Nombre de la especie.

- **Gen:**

- id_gen: Identificador primario del gen.
- nombre: Nombre del gen.
- id_proteina: Foreign key que apunta a la proteína que codifica.

- **Referencia:**

- id_referencia: Identificador primario de la referencia.
- año: Año de publicación.
- titulo: Titulo de la publicación.
- id_proteina: Foreign key que apunta a la proteína a la cual describe.

- **Autor:**

- id.autor: Identificador primario del autor
- Nombre: Nombre y apellido del autor.

2.2.3 Diagrama de tablas

Habiendo desarrollado el diseño, el diagrama de tablas será el siguiente:

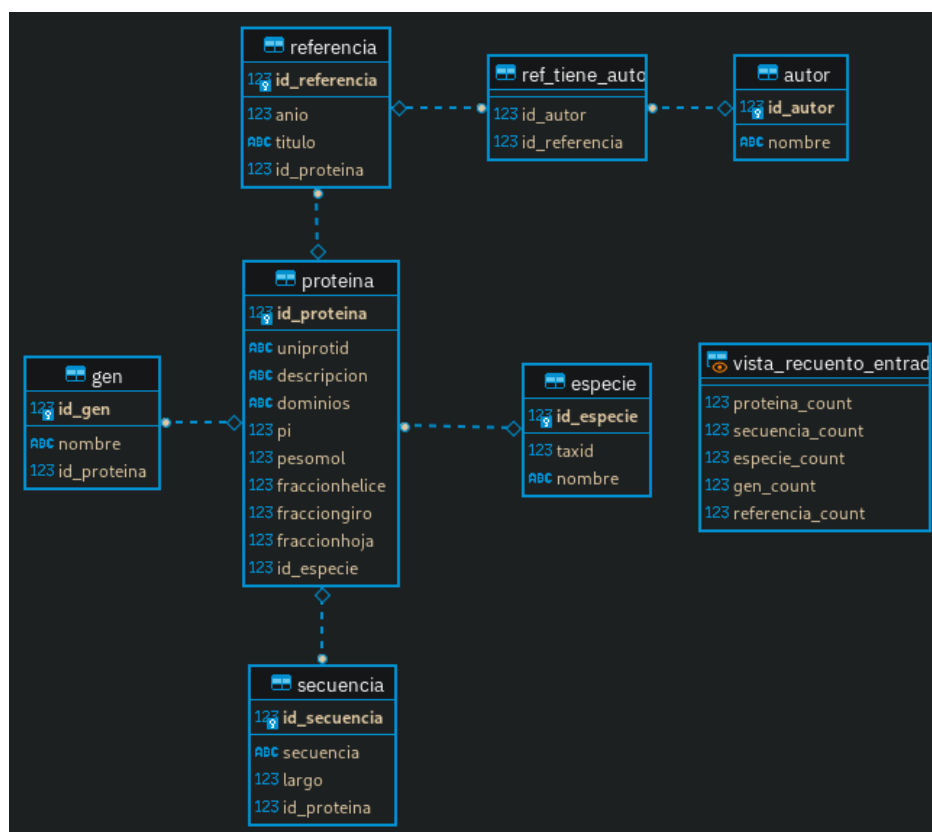


Fig. 6: Diagrama de tablas.

En la figura 6 se puede observar la presencia de una vista, esta será desarrollada más adelante.

2.3 Implementación

Para llevar a cabo la implementación de la base de datos, se optó por utilizar PostgreSQL como el sistema de gestión de bases de datos. Esta decisión se tomó considerando que fue el utilizado durante el cursado de la materia y es uno de los más distribuidos en el campo de las bases de datos.

2.3.1 Declaración

Teniendo ya trabajado el diseño de la base de datos, llevé a cabo la creación de las tablas con sus atributos y las restricciones necesarias. Para ello tuve en cuenta los tipos de datos admitidos por PostgreSQL y las características de los datos que había obtenido. Teniendo en cuenta que muchos de los datos que me interesaba registrar eran cadenas de caracteres, tuve que tomar decisión importantes de largos predefinidos, truncando de ser necesario, y en otros casos, como el de la secuencia ya mencionada, tomé la decisión de dejarlo indefinido.

Una vez que tuve claro cómo debían estructurarse las tablas, creé un archivo [.sql](#) que contenía las sentencias de lenguaje de definición de datos (DDL) necesarias para reflejar el diseño propuesto.

2.3.2 Adquisición e inserción de datos

Como desarrolle previamente, algunos de los datos de relevancia fueron extraídos del archivo CSV adquirido en el sitio web de la base de datos de la Universidad de Toronto. Luego con

estos datos, realicé los pedidos correspondientes a la API de UniProt y extraje información de la secuencia con el modulo ProtParam del cual se habló previamente, almacenando ambas salidas en archivos CSV.

Pueden examinarse los algoritmos utilizados en el repositorio de GitHub asociado a este trabajo para obtener una visión más detallada de su funcionamiento.

Luego trabaje con estos archivos en DataFrames de pandas, con el propósito de preparar la información que deseaba insertar en las respectivas tablas. Realicé una serie de operaciones sobre estas estructuras de datos, aplicando conceptos de álgebra relacional que había adquirido en la materia, como los joins, para lograr esto.

Habiendo obtenido en forma de DataFrame todos los datos que quería representar, utilicé la función *to_sql(...)* de pandas, que me permitió realizar automáticamente la conexión con la base de datos y la inserción de los datos en las tablas que había declarado previamente en la sección anterior.

2.4 Manipulación

2.4.1 Consultas requeridas

En primera instancia se requería que se haga uso de 3 consultas avanzadas para la manipulación de los datos. Por ello es que planteo tres posibles casos, realicé las consultas y los imprimí

Primer consulta

En la primera de ellas supuse que se deseaba conocer el número de proteínas asociadas a cada especie de las cuales se tenía registro.

```
-- 1) Número de proteína por especie.  
SELECT COUNT(prot.*) AS cnt, esp.Nombre  
FROM proteina as prot  
JOIN especie as esp  
    ON esp.id_especie = prot.id_especie  
GROUP BY esp.Nombre  
ORDER BY cnt DESC;
```

Fig. 7: Primer consulta

Como se puede apreciar en la figura 7 esta me permitió hacer uso de funciones de agregación, realizar agrupación y utilizar más de una tabla. El resultado obtenido es la tabla siguiente:

Total	Especie
390	Homo sapiens
339	Mus musculus
167	Caenorhabditis elegans
73	Drosophila melanogaster
1	Danio rerio

Tabla 1: Salida de la primer consulta.

Segunda consulta

Para esta consulta planteo una consulta para obtener la proteína presente en todas las especies, ya que me iba a permitir hacer uso de la división con la doble negación de

```
-- 2) Mostrar la proteína que esta en todas las especies.  
SELECT DISTINCT prot.uniprotid  
FROM proteina as prot  
WHERE NOT EXISTS (  
    SELECT esp.id_especie  
    FROM especie as esp  
    WHERE NOT EXISTS (SELECT *  
        FROM proteina as prot2  
        WHERE esp.id_especie = prot2.id_especie AND  
            prot.uniprotid = prot2.uniprotid));
```

Fig. 8: Segunda consulta.

existencia.

Al ejecutarla, como era de esperarse al entender la situación de la vida real que buscamos registrar, no devolvió ninguna coincidencia.

Tercer consulta

Aprovechando lo obtenido en la primer consulta, me propuse encontrar los títulos de las referencias que hablen sobre la única proteína de la especie *Danio rerio*.

```
-- 3) Encontrar el titulo de la/s referencia/s que describe la única proteína  
-- que se tiene registro en Danio rerio  
SELECT rf.titulo  
FROM referencia as rf  
JOIN (SELECT prot.uniprotid, prot.id_proteina  
    FROM proteina as prot  
    JOIN (SELECT esp.id_especie  
        FROM especie as esp  
        WHERE esp.Nombre = 'Danio rerio') as A  
    ON prot.id_especie = A.id_especie) as B  
ON rf.id_proteina = B.id_proteina;
```

Fig. 9: Tercer consulta.

Tras ejecutar la consulta, obtuve la siguiente tabla:

	Título
0	Vegetal localization of the maternal mRNA encoding an EDEN-BP/Bruno-like protein in zebrafish.
1	Regulation of alternative splicing of alpha-actinin transcript by Bruno-like proteins.

Tabla 2: Salida de la tercer consulta.

Luego, se requería crear una consulta que utilice tres tablas, contenga una condición de igualdad y una condición de rango. Además debía escribirse el árbol de ejecución de la consulta en álgebra relación y optimizarlo para escribir la consulta optimizada. La consulta propuesta es la siguiente:

```
SELECT prot.descripcion as Nombre, sec.secuencia as Secuencia
FROM proteina as prot
  JOIN secuencia as sec
    ON sec.id_proteina = prot.id_proteina
  JOIN especie as esp
    ON prot.id_especie = esp.id_especie
WHERE esp.taxId = 9606 AND prot.pi BETWEEN 6.4 AND 13.2;
```

Fig. 10: Consulta no optimizada.

De analizar esta consulta, pude conformar el siguiente árbol de ejecución (a la izquierda se lo expresa con los productos y selecciones por separado y a la derecha con los joins identificados):

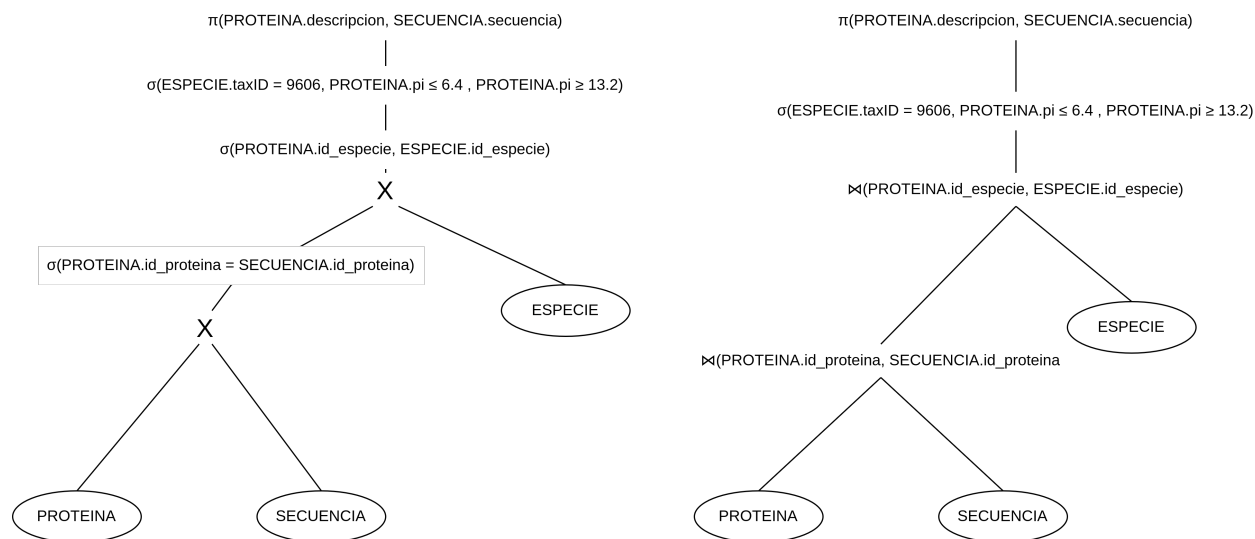


Fig. 11: Árbol de ejecución no optimizado

A partir de este, puedo hacer uso de la heurística para la optimización de la consulta. Siguiendo esta lógica pude obtener el siguiente árbol de ejecución no optimizado:

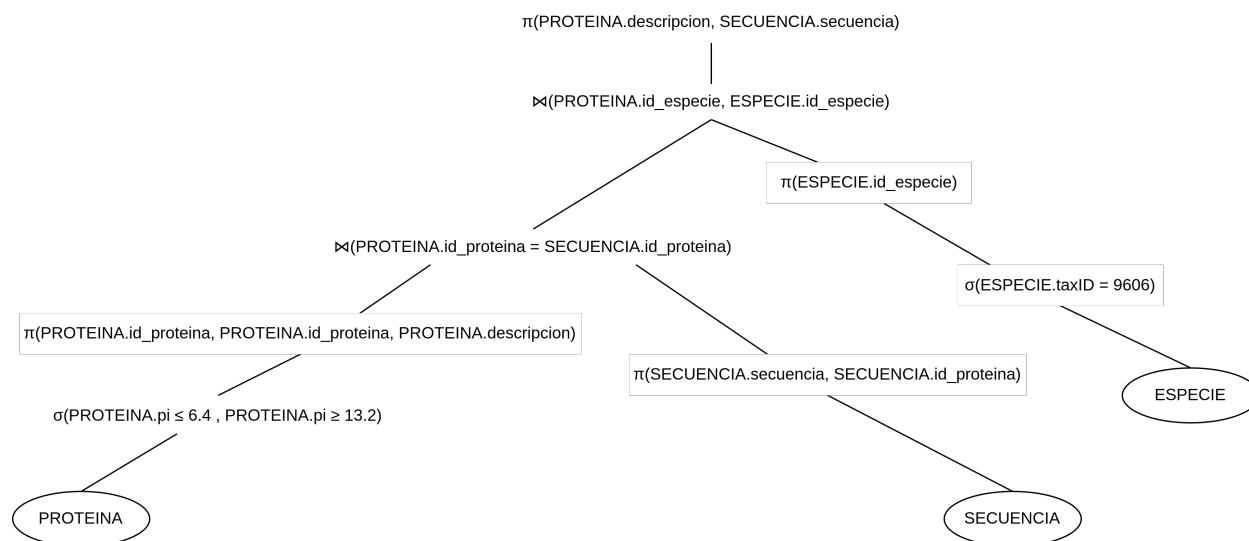


Fig. 12: Árbol optimizado

Con el árbol ya optimizado, fue una excelente guía para la optimización de la consulta. Esta quedó de la siguiente manera:

```
-- Optimizada:
SELECT prot.descripcion as Nombre, sec.secuencia as Secuencia
FROM (
  SELECT esp.id_especie
  FROM especie as esp
  WHERE esp.taxid = 9606
) as A
JOIN (
  SELECT prot.descripcion, sec.secuencia, prot.id_especie
  FROM (
    SELECT prot.id_proteina, prot.descripcion, prot.id_especie
    FROM proteina as prot
    WHERE prot.pi BETWEEN 6.4 AND 13.2
  ) as B
  JOIN (
    SELECT sec.secuencia, sec.id_proteina
    FROM secuencia as sec
  ) as C
  ON C.id_proteina = B.id_proteina
) as D
ON A.id_especie = D.id_especie
```

Fig. 13: Consulta optimizada

Ambas consultas pudieron ser ejecutadas y se observó el mismo número de filas.

2.4.2 Uso de interfaz de acceso a datos

Para la generación de valor agregado a partir del uso de una interfaz de acceso a datos me propuse desarrollar una aplicación web que presentase al usuario información extraída de la base de datos de forma apropiada.

Con el objetivo de generar un valor agregado a través de una interfaz de acceso a datos,

me propuse desarrollar una aplicación web diseñada para presentar de manera eficiente la información extraída de la base de datos. Para llevar a cabo este proyecto, opté por utilizar Dash [7], una herramienta que me permitió escribir en Python, con nociones de HTML y CSS, la aplicación que tenía en mente. Esta elección facilitó la utilización de los conectores que habíamos abordado en clases para interactuar con la base de datos, además de proporcionarme potentes herramientas para la presentación de datos.

El diseño de mi aplicación se estructuró en tres páginas distintas, lo que me permitió aprovechar y acceder a las diversas tablas de mi base de datos:

- Información sobre el número de entradas para las diferentes tablas. [14]
- Una donde el usuario pueda seleccionar una proteína y realizar las siguientes acciones [15]:
 - Visualizar su secuencia, realizar búsquedas en ella con patrones regulares y guardarla en formato *fasta*.
 - Acceder a la página de UniProt de la proteína y realizar búsquedas en GenBank para el gen asociado a ella.
 - Visualizar su descripción, dominios, punto isoeléctrico, peso molecular y las fracciones de cada estructura secundaria que le corresponden.
- Una última página donde el usuario pueda visualizar una tabla donde se informen el número de publicaciones que tiene cada autor y realizar búsquedas. [16]

Como dije, declaré una vista que contenga los recuentos de entradas en las diferentes tablas para que el acceso a esta información sea más fácil y para la aplicación específica, además de ser una buena oportunidad de poner en práctica lo trabajado durante la cursada.

Finalmente logré obtener un resultado satisfactorio para proveer al usuario de una interfaz para explorar la base de datos. Las distintas páginas desarrolladas fueron las siguientes:

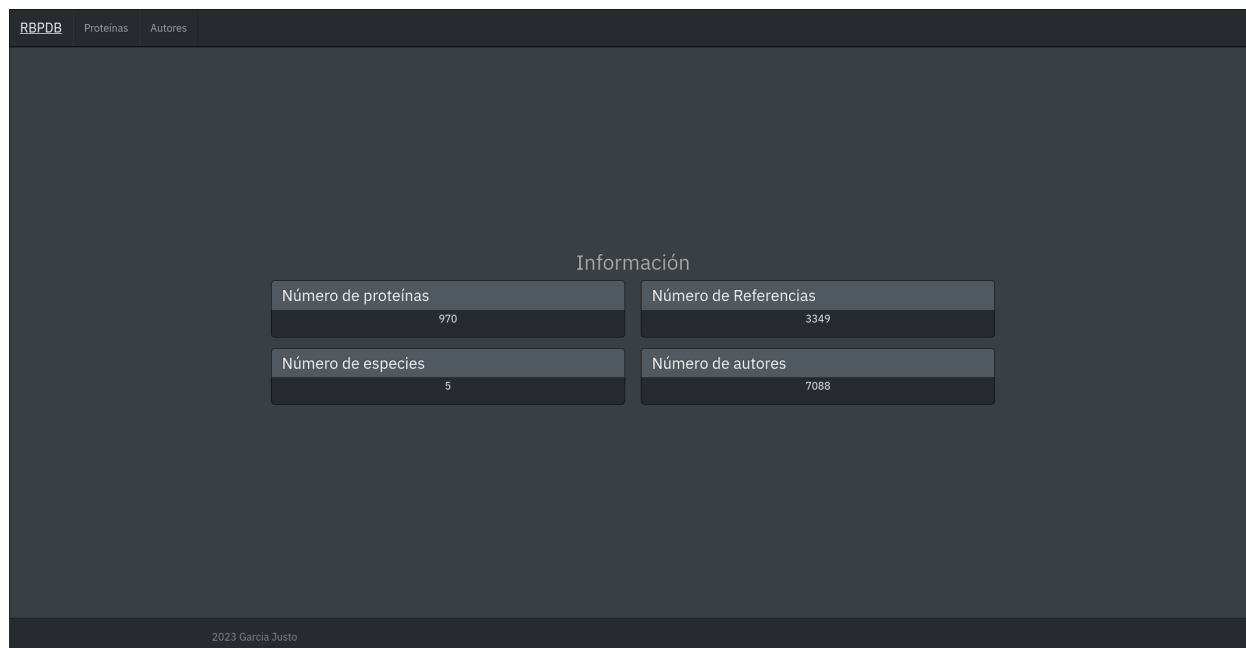


Fig. 14: Página 1

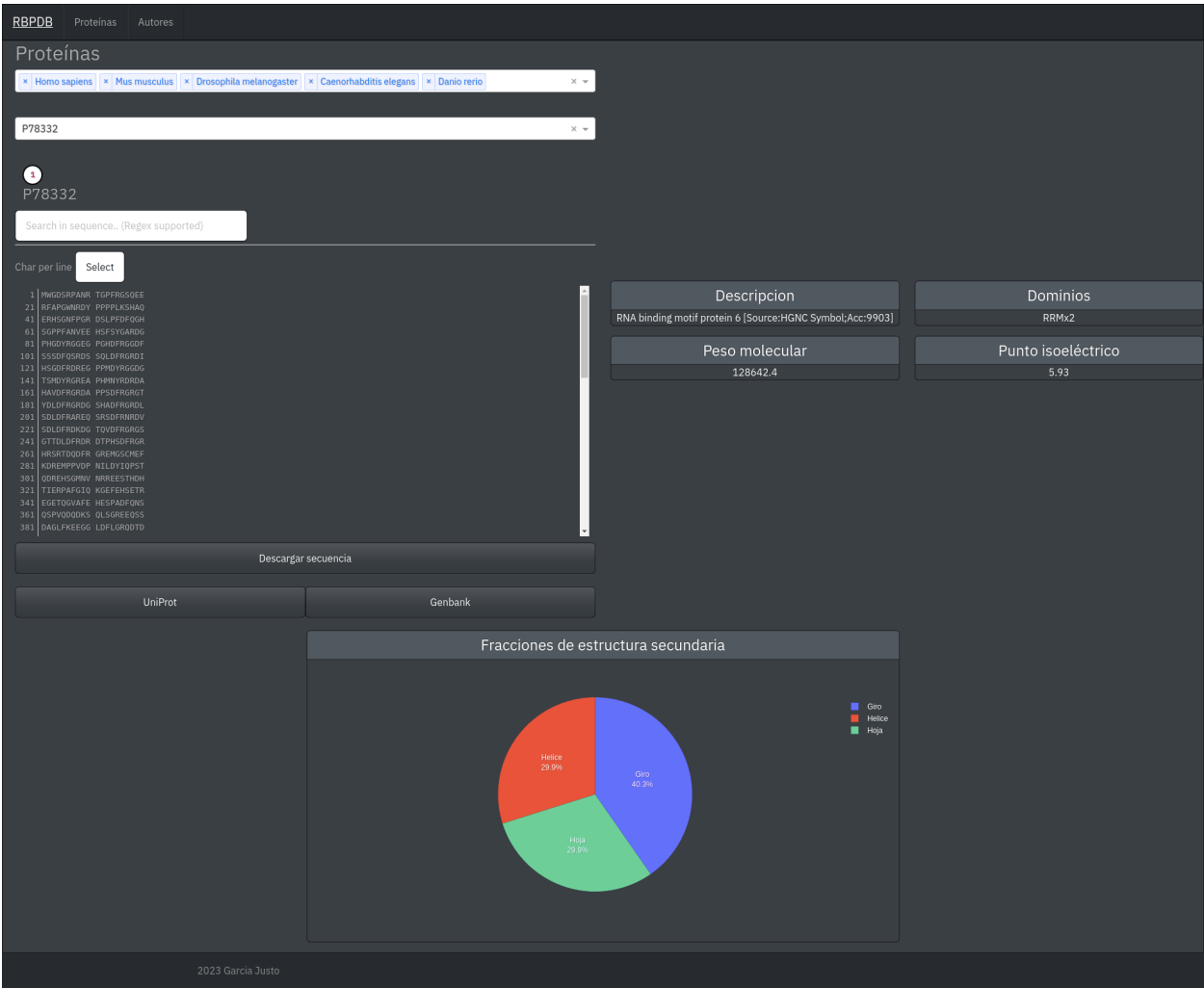


Fig. 15: Página 2

RBPDB Proteínas Autores

autor	totalreferencias
filter data...	
Ohara O.	548
Nakamura Y.	485
Yamazaki M.	399
Watanabe M.	399
Nagase T.	389
Eichler E.E.	278
Liu J.	264
Kitamura H.	259
Schneider C.	253
Mattick J.S.	253

1 / 700

2023 Garcia Justo

Fig. 16: Página 3

3 Conclusión

Con el desarrollo de este Trabajo Integrador Final se presentó la oportunidad de aplicar y consolidar los conocimientos adquiridos a lo largo del curso de bases de datos. Fui capaz de diseñar, implementar y utilizar una base de datos de manera autónoma.

Además, es importante destacar la libertad concedida en la elección de herramientas, temas y enfoques. Permitiendo abordar temas de interés personal, explorar nuevas herramientas, exigirse a uno mismo y fomentar la creatividad y diversidad.

Referencias

- [1] Justo Garcia. *justog220/TIF-BD: Trabajo integrador final de base de datos*. URL: <https://github.com/justog220/TIF-BD> (visited on 10/25/2023).
- [2] Kate B. Cook et al. “RBPDB: a database of RNA-binding specificities”. In: *Nucleic Acids Research* 39 (suppl.1 Jan. 1, 2011), pp. D301–D308. ISSN: 0305-1048. DOI: [10.1093/nar/gkq1069](https://doi.org/10.1093/nar/gkq1069). URL: <https://doi.org/10.1093/nar/gkq1069> (visited on 09/26/2023).
- [3] *Postman API Platform — Sign Up for Free*. Postman. URL: <https://www.postman.com> (visited on 10/25/2023).
- [4] Biopython. *Biopython*. Biopython. URL: <https://biopython.org/> (visited on 10/25/2023).
- [5] Elisabeth Gasteiger et al. “Protein Identification and Analysis Tools on the ExPASy Server”. In: *The Proteomics Protocols Handbook*. Ed. by John M. Walker. Springer Protocols Handbooks. Totowa, NJ: Humana Press, 2005, pp. 571–607. ISBN: 978-1-59259-890-8. DOI: [10.1385/1-59259-890-0:571](https://doi.org/10.1385/1-59259-890-0:571). URL: <https://doi.org/10.1385/1-59259-890-0:571> (visited on 10/25/2023).
- [6] *pandas documentation — pandas 2.1.1 documentation*. URL: <https://pandas.pydata.org/docs/index.html> (visited on 10/26/2023).
- [7] *Dash Documentation & User Guide — Plotly*. URL: <https://dash.plotly.com/> (visited on 10/26/2023).