FACULTAD DE CIENCIAS AGRARIAS

UNIVERSIDAD NACIONAL DE ROSARIO

“EURECA (Eukaryote DNA Repair Capacity) una plataforma web con una base de datos sobre sistemas de reparación indirecta del ADN en eucariotas y con herramientas bioinformáticas”

Michelle Christine CHIRINOS ARIAS

TRABAJO FINAL PARA OPTAR AL TÍTULO DE ESPECIALISTA EN

Bioinformática

DIRECTORA: Marcela Claudia Dotto

2021

“EURECA (Eukaryote DNA Repair Capacity) una plataforma web con una base de datos sobre sistemas de reparación indirecta del ADN en eucariotas y con herramientas bioinformáticas”

Michelle Christine Chirinos Arias

Bióloga – Universidad Nacional Agraria La Molina (Lima, Perú)

Este Trabajo Final es presentado como parte de los requisitos para optar al grado académico de Especialista en Bioinformática de la Universidad Nacional de Rosario y no ha sido previamente presentado para la obtención de otro título en esta u otra Universidad. El mismo contiene los resultados obtenidos en investigaciones llevadas a cabo durante el período comprendido entre 2018-2020 bajo la dirección de la Dra. Marcela Dotto.

Autora: Michelle C. Chirinos Arias

Directora: Marcela Claudia Dotto

Defendida: 16 de Mayo del 2022.

AGRADECIMIENTOS

A Claudia Spampinato por la constante ayuda y la confianza para permitirme desarrollar este Postgrado de Especialización en Bioinformática. A mi familia: Ivonne, José (que ya está al lado de Dios) y Lucía, por apoyarme en todo momento y escuchar todas las historias del grupo de bioinformáticos. A los bioinformáticos Pablo Vélez y Daniel Souza, por ser mis compañeros en todo el cursado. Un agradecimiento especial a las chicas de mi laboratorio Valentina Gonzalez, Romina Rosati, Rocío Ramos y Luciana Lario por ayudarme sobre todo cuando cursar se me complicaba con las cosas del laboratorio y por todos los mates compartidos.

¡Muchas gracias a todos!

A mi familia en Perú, en
especial en memoria de mi
papito que descansa con Dios

ABREVIATURAS Y CONCEPTOS

ADN	Ácido desoxirribonucleico
ARN	Ácido ribonucleico
BER	Reparación por escisión de bases
BD	Base de datos
BLOB	Binary Large Object u objetos binarios grandes
CGI	Common Gateway Interface o interfaz de entrada común
CHAR	Cadena de caracteres
CRUD	Crear, leer, actualizar y borrar
CRUDL	Crear, leer, actualizar, borrar y listar
CSS	Cascading Style Sheets u hojas de estilo en cascada
ERO	Especies reactivas de oxígeno
EURECA	Eukaryote Repair Capacity
FASTA	Formato de fichero informático basado en texto
FK	Foreign Key o clave foránea
FN	Forma Normal
FTP	File Transfer Protocol o protocolo de transferencia de archivos
HTML	HyperText Markup Language
ID	Código de identificación
INT	Integer o entero
MAMP	Conjunto de programas usados para desarrollar sitios web dinámicos sobre sistemas Macintosh
MLH	Homólogo a MutL

MMR	Mismatch Repair o reparación de apareamientos incorrectos
MSH	Homólogo de MutS
MSI	Inestabilidad de microsatélites
NCBI	National Center for Biotechnology Information
NER	Reparación por escisión de nucleótidos
PCR	Reacción en Cadena de la Polimerasa
PMS	Postmeiotic segregation
PHP	PreProcesador de Hipertexto
PK	Principal Key o clave primaria
RGB	Red, Green, Blue (composición del color)
RNAseq	Secuenciación masiva del ARN
Sitios AP	Sitios apurínicos-apirimidínicos
SQL	Structure Query Language
SSR	Single Simple Repeats o microsatélites
TAIR	The Arabidopsis Information Resource
UI	Interfaz de Usuario
URL	Localizador de recursos uniforme
UV	Ultravioleta
UX	Experiencia usuario
VARCHAR	Cadena de caracteres de longitud variable
WAMP	Windows, Apache, MySql y PHP
WT	Wild Type o salvaje
XAMPP	Software con sistema de gestión de base de datos MySQL, Apache, PHP y Perl.

RESUMEN

Los organismos vivos nos encontramos continuamente expuestos a factores bióticos y/o abióticos que pueden producir daño en el ADN; sin embargo, existen mecanismos para repararlo. Dentro de ellos, se encuentran los mecanismos de reparación indirecta que incluyen a los sistemas BER (reparación por escisión de bases), NER (reparación por escisión de nucleótidos) y MMR (reparación de apareamiento incorrecto). Existe una gran cantidad de publicaciones e información disponible sobre estos mecanismos para diversas especies, especialmente para humanos y en menor medida para plantas. Sin embargo, los datos se encuentran dispersos y es necesario organizar y centralizarlos en una plataforma disponible para la comunidad científica. Con este objetivo se planteó crear a EURECA (**Eu**karyote DNA **Re**pair **Ca**pacity), una plataforma web gratuita que incluye una base de datos con herramientas bioinformáticas básicas, pero útiles para el estudio de los sistemas de reparación indirectos del ADN de eucariotas, lo que facilitará el acceso a la información existente y por consiguiente, su estudio.

ABSTRACT

Living organisms are continuously exposed to either by biotic or abiotic factors that can cause DNA damage; however, there are mechanisms to repair it. These include the indirect repair mechanisms including the BER (Base Excision Repair), NER (Nucleotide Excision Repair) and MMR (Mismatch Repair) systems. There is a large amount of literature and information available regarding these mechanisms in various species, particularly in humans and, to a lesser extent, in plants. However, the data is scattered and needs to be organized and centralized in a platform available to the scientific community. For this purpose EURECA (Eukaryote DNA Repair Capacity) was created; it is a free web platform that includes a database with basic useful bioinformatics tools to study the indirect DNA repair systems of eukaryotes, which will facilitate access to existing information and its study.

ÍNDICE

1. INTRODUCCIÓN

Los organismos vivos estamos constantemente expuestos a diversos factores abióticos como la salinidad, sequía, temperaturas extremas, radiación UV, metales pesados y a factores bióticos como los microorganismos patogénicos, entre otros. Estos factores pueden afectar, de manera directa o indirecta, no sólo la integridad de los componentes estructurales y enzimáticos de la célula, tales como los lípidos y las proteínas, sino también al ADN, mediante la formación de especies reactivas del oxígeno (EROs) (Iyama & Wilson, 2013; Tafurt y col., 2014). Por esta razón, todos los organismos vivos, poseen mecanismos de reparación del ADN que les permiten mantener la estabilidad de su genoma.

1.1. Sistemas de reparación del daño en el ADN

La reparación del ADN es un proceso complicado que involucra muchos sistemas. Dentro de ellos, se encuentran los mecanismos de reparación directa o también conocidos como reversión de la lesión, que incluye a los sistemas de fotorreactivación, alquiltransferencia y demetilación oxidativa. Mientras que, los mecanismos de reparación indirecta, en los cuales nos centraremos en este trabajo, incluyen a los sistemas BER (por sus siglas en inglés "*Base Excision Repair*"), NER (por sus siglas en inglés "*Nucleotide Excision Repair*") y MMR (por sus siglas en inglés "*Mismatch Repair*") (Spampinato y col., 2009; Iyama & Wilson, 2013; Kunkel, 2015).

El sistema BER corrige el daño al ADN producido por la oxidación, deaminación y alquilación. Estas lesiones causan una pequeña distorsión de la estructura de hélice del ADN y aunque pueden ocasionarse de forma espontánea, también pueden ser causadas por los químicos ambientales, la radiación o el tratamiento con drogas citostáticas. Si este daño no se repara a tiempo, puede llegar a ser citotóxico y genotóxico (Hoeijmakers y col., 2001); por lo que el sistema BER protege contra el cáncer, el envejecimiento y la neurodegeneración que puede ocurrir sobre el ADN que se encuentra tanto en

el núcleo como en la mitocondria (Krokan y Bjoras y col., 2013). La reparación inicia con el reconocimiento y la remoción de la base dañada que realizan las ADN glicosilasas (Krokan y Bjoras., 2013) generando sitios apurínicos o apirimidínicos (sitios AP). Los sitios AP generados son reconocidos por una AP endonucleasa de tipo II que elimina el resto de nucleótido por hidrólisis y posteriormente una exonucleasa termina el corte, dejando el sitio vacío para que la polimerasa añada el nuevo nucleótido y por último se integre a la cadena de ADN por la ligasa (Tafurt y col., 2014).

El sistema NER repara el daño en el ADN causado por la radiación UV, agentes mutagénicos, quimioterapia, etc (Leibeling y col., 2006; Tafurt y col., 2014). En el caso de procariotas son cuatro proteínas (UvrA, UvrB, UvrC y UvrD) las encargadas de reconocer el daño en el ADN (específicamente la distorsión de la doble hélice), mientras que en mamíferos son más de treinta (Tafurt y col., 2014). Los pasos para la reparación por el sistema NER son similares tanto en organismos simples como las bacterias, como en los más complejos como los mamíferos y las plantas. Los pasos incluyen el reconocimiento del nucleótido dañado o de estructuras que interrumpen la estructura de la doble hélice, la remoción del oligonucleótido que incluye al nucleótido dañado, la síntesis, reparación y ligación (Spivak, 2015). Deficiencias en estas vías de reparación están relacionadas con afecciones como el síndrome de Cockayne (CS), el xeroderma pigmentoso (XP) y con el cáncer e inestabilidad genómica (Hoeijmakers y col., 2001; Tafurt y col., 2014).

El sistema MMR funciona principalmente en la corrección de errores biosintéticos producidos durante la replicación o durante la recombinación, aunque también es capaz de reconocer lesiones en el ADN inducidas por factores endógenos y exógenos (Tafurt y col., 2014). El sistema se encuentra evolutivamente conservado y ampliamente distribuido tanto en organismos procariotas como eucariotas y dada su importancia es tema de revisión frecuente (Jiricny 2013; Erie and Weninger 2014; Kunkel y Erie 2015; Friedhoff y col., 2016 Li y col., 2016). Las proteínas del tipo MutS de procariotas se

conocen como MSH en eucariotas (homólogos de MutS); mientras que las proteínas del tipo MutL se conocen como MLH (homólogos de MutL) y PMS (cuyo nombre refleja que los mutantes *pms* exhiben altos niveles de segregación postmeiótica) y se designan como MLH1-MLH3 y PMS1-PMS2. Una característica especial de las plantas es la presencia de una proteína adicional denominada MSH7 (Spampinato y col., 2009; Tam y col., 2009; Tafurt y col., 2014; Chirinos-Arias y Spampinato, 2020; Chirinos-Arias y Spampinato, 2021). Todos los miembros de las proteínas descriptas forman heterodímeros funcionales, donde sólo algunos son responsables de mantener la estabilidad del ADN nuclear durante la división mitótica. Dichos heterodímeros comprenden a MutSα (MSH2-MSH6), MutSβ (MSH2-MSH3), MutSγ (heterodímero exclusivo de plantas, MSH2-MS7) y MutLα (MLH1-PMS1 en las levaduras y MLH1-PMS2 en las células humanas) (Gómez and Spampinato, 2013; Jiricny, 2013; Erie and Weninger, 2014; Tafurt y col., 2014; Kunkel and Erie, 2015; Friedhoff y col., 2016).

1.2. Bases de datos en investigación científica

Como consecuencia de la gran expansión en la generación de información por parte de la investigación científica a nivel mundial, el uso de bases de datos (BD) se ha constituido en un recurso muy importante para la comunidad con el correr de los años. Las BD permiten la definición y almacenamiento de la información de forma ordenada, y los sistemas manejadores de bases de datos proveen los mecanismos para una gestión eficiente de su contenido, de esta manera se consigue que no haya datos redundantes y se ahorra espacio en el almacenamiento.

En el caso de los datos biológicos, uno de los problemas más frecuentes es que un solo gen puede tener varios nombres, lo que genera confusión y hace que algunos datos se encuentran dispersos, por lo que es difícil hallarlos y relacionarlos. Existen distintos tipos de BD, dentro de ellas las bases de datos relacionales son el modelo más utilizado, y lo son porque permiten

establecer relaciones entre los datos. Este tipo de BD se compone de varias tablas o relaciones, donde cada tabla tiene su propio conjunto de registros y no pueden existir dos tablas con el mismo nombre, entre otras características (Camps y col., 2005; Marqués, 2011; Millán, 2017).

Existe una gran cantidad de publicaciones sobre mecanismos de reparación de ADN en humanos y levaduras (Li y col., 1995; Mellón y col., 1996; Zhang y col., 2000; Groth y col., 2005; Kadyrov y col., 2006). Caso contrario ocurre con las plantas, donde la mayoría de trabajos se concentra en *Arabidopsis thaliana* (Bilichak y col., 2012; Gómez & Spampinato, 2015; Lario y col., 2015, Leonard y col., 2003), aunque existen investigaciones que conciernen a otras especies como alfalfa (Quaite y col., 1994), trigo (Taylor y col., 1996), pepino (Takeuchi y col., 1998), espinaca (Yoshihara y col., 2005), cebada (Lloyd y col., 2007; Manova y col., 2009), soja (Yamamoto y col., 2008), arroz (Hirouchi y col., 2003), maíz (Stapleton y col., 1997; Liu y col., 2019), etc. En la actualidad no existe una BD que centralice la información relacionada con los mecanismos de reparación del ADN, es por ello que es necesario ordenar los datos disponibles para poder buscar fácilmente los trabajos ya realizados y no redundar en las investigaciones y así lograr una gestión más eficiente de los datos existentes. Respondiendo a esta necesidad, se creó una plataforma web online y gratuita que incluye una base de datos y herramientas bioinformáticas básicas (scripts automáticos de análisis de información bioinformática) que facilitarán el estudio del sistema de reparación indirecta del ADN en eucariotas. Por ejemplo, en esta plataforma es posible realizar búsquedas de motivos microsatélites o *Simple Sequence Repeats* (SSR), las cuales están relacionados a inestabilidad genómica, una característica del sistema de reparación del ADN que funciona erróneamente. También posee una herramienta que facilita el análisis de vectores en fase con el marco abierto de lectura de un gen de interés, ya que las técnicas de clonado son muy usadas en este campo, junto con una herramienta que facilita el análisis de sitios de restricción en secuencias de interés de forma automática.

2. OBJETIVOS

2.1. General:

- Crear una plataforma web con una base de datos relacional del sistema de reparación indirecta del ADN para organismos eucariotas (levaduras, humanos y plantas) llamada EURECA, integrando y ordenando la información disponible concerniente a proteínas, genes, factores bióticos y abióticos, fenotipo y artículos científicos.

2.2. Secundarios:

- Crear una página web para EURECA
- Facilitar la búsqueda de motivos microsatélites o Simple Sequence Repeats (SSR) mediante la creación de una herramienta bioinformática
- Facilitar el análisis de vectores en fase
- Facilitar el análisis de sitios de reconocimiento de enzimas de restricción

3. MATERIALES Y MÉTODOS

3.1. Elección de los datos de las tablas

Se armaron las tablas preliminares con los datos necesarios para crear la base de datos dentro de EURECA. Cada tabla fue identificada con un título y en cada columna se colocaron los atributos, se identificó el tipo de dato y el dominio.

3.2. Normalización del modelo relacional

Se identificó el tipo de atributo de cada elemento de la tabla: cuando el atributo no fue monovaluado y simple, se desglosó hasta que lo fuera, ampliando de esta forma las columnas de las tablas. Se identificaron las claves primarias (PK) de cada una de las tablas preliminares, se verificó que las tablas cumplan las 3 formas normales y se realizaron las modificaciones pertinentes si no lo cumplían. Luego se reemplazaron los nombres largos de cada atributo, por nombres más cortos y en inglés que son los que finalmente se colocaron en las tablas. En primera instancia no se completaron las tuplas (con el valor de las entidades o datos) para hacer más sencillo el diseño de la estructura de la base de datos (BD). Por último, se agregaron las claves foráneas (FK) y se construyó el modelo relacional con las tablas permanentes.

3.3. Recolección, procesamiento y curación de datos

Se buscaron artículos científicos de temas concernientes a los diferentes sistemas indirectos de reparación de ADN. Se utilizaron artículos que estuvieran indexados a bases de datos científicas de prestigio como Web of Science, Scopus y Scielo. Se realizaron búsquedas para tres especies: *Arabidopsis thaliana* (planta), *Homo sapiens* (humano) y *Saccharomyces cerevisiae* (levadura). Cabe destacar que la base especializada para *Arabidopsis* es el TAIR, disponible en https://www.arabidopsis.org/ y para levaduras es YeastGenome, disponible en https://www.yeastgenome.org/), las cuales fueron de gran utilidad para encontrar investigaciones publicadas enfocadas en los mecanismos de reparación indirecta MMR, BER y NER.

Utilizando estos artículos científicos se pudo completar las tablas correspondientes a publicaciones, factores bióticos y abióticos.

Para recopilar la información relacionada con genes, se buscaron los códigos de cada uno de los genes relacionados a los tres sistemas de reparación indirecta del ADN en el NCBI (disponible en https://www.ncbi.nlm.nih.gov/), así como los diferentes nombres con los cuales son conocidos, su función y su secuencia en formato FASTA. Por otro lado, los códigos de identificación de las proteínas, así como sus nombres alternativos, peso molecular y número de aminoácidos fueron obtenidos de la base de datos de Uniprot (disponible en https://www.uniprot.org/). Las características de las proteínas como punto isoeléctrico, índice de inestabilidad, índice alifático y GRAVY fueron determinados con la herramienta ProtParam del ExPASy (disponible en https://web.expasy.org/protparam/) (Gasteiger, 2005). Cada dato se agregó en la tabla normalizada correspondiente utilizando el programa Excel. Todos los datos fueron curados cuidadosamente y almacenados en la base de datos dentro de la plataforma web EURECA.

3.4. Esquematización de la página web

Se definió la arquitectura de la información, el árbol de contenidos, el prototipo de la página web mediante un diagrama a manera de mapa conceptual. Para esto se empleó una aplicación o programa de boceto llamada Gliffy, disponible en https://www.gliffy.com/

3.5. Diseño y programación de la página web

Se usó el lenguaje de marcado HTML (*HyperText Markup Language*) para el desarrollo de la página web y se la hizo responsiva (adaptable a diferentes navegadores y dispositivos). Mientras que se usaron CSS (*Cascading Style Sheets* u hojas de estilo en cascada) para darle estilo y una mejor presentación, y el lenguaje de programación utilizado para el lado del usuario fue JavaScript que permite darle interactividad dinámica al sitio web. En todos los casos se usó el editor de texto Notepad++ que es de código fuente libre con soporte para varios lenguajes de programación. Las imágenes

fueron realizadas exclusivamente para este trabajo, para evitar cualquier conflicto por derechos de autoría. Para ello, se diseñaron y vectorizaron las imágenes con el programa Adobe Illustrator CC 2015, guardándolas en formato .png y luego fueron usadas en la confección de la página web. Cabe aclarar que la página web se escribió en inglés, para que tenga mayor acogida por la comunidad científica.

3.6. Programación e implementación de las herramientas bioinformáticas

Se implementaron 3 herramientas bioinformáticas útiles para biólogos moleculares que investigan los diferentes mecanismos de reparación de ADN:

A) *Búsqueda de motivos microsatélites*: la herramienta bioinformática desarrollada permite identificar los motivos microsatélites en una secuencia de ADN, para posteriormente diseñar cebadores específicos y poder comparar la estabilidad genética en la especie a estudiar. Para ello se debe proporcionar la siguiente información como *input*: la secuencia de ADN; los motivos (desde dinucleótidos hasta decanucleótidos) y el número de repeticiones a buscar. La herramienta proporciona como *output* la identificación de los microsatélites presentes en la secuencia de ADN ingresada.

B) *Vector en fase*: el *script* automático creado permite colocar una secuencia de ADN como *input* y genera como *output* la secuencia separada cada 3 bases (lo que permite ver si la secuencia está en fase o no).

C) Análisis de sitios de restricción: la herramienta bioinformática creada puede encontrar sitios de reconocimiento de enzimas de restricción en una secuencia de ADN y mostrar el nombre de dichas enzimas junto con los fragmentos cortados. De esta forma, se puede elegir qué enzimas usar para generar fragmentos romos o cohesivos en un gen de interés, sino que además se

puede verificar si las enzimas seleccionadas pueden generan cortes no deseados dentro del gen analizado.

Para crear las tres herramientas bioinformáticas se usaron scripts en JavaScript y se los incrustaron en el código HTML de la página web. Para mejorar la interactividad se instaló la librería JsQuery disponible en https://jquery.com/download/. Los nombres y cortes de las enzimas de restricción se obtuvieron de https://enzymefinder.neb.com/#!/#nebheader. Los algoritmos para cada herramienta se representaron en diagramas de flujo usando la herramienta https://www.lucidchart.com

3.7. Instalación del servidor local

Se instaló el servidor web de Apache XAMPP (disponible en https://www.apachefriends.org/es/index.html) y todos los scripts de la página web fueron guardados en XAMPP > xamppfiles > htdocs > dashboard > eurecaweb dentro de la computadora local y fueron consultados ingresando a localhost/dashboard/eurecaweb/nombre_del_archivo.php

3.8. Construcción de la base de datos

Los datos tomados en el apartado 3.3 fueron guardados en diferentes tablas en formato .xlsx. Se usó el lenguaje de programación SQL (*Structured Query Language*) para administrar y recuperar información del sistema de gestión de la base de datos relacional dentro de la plataforma web EURECA, mediante MySql (https://www.mysql.com/). Se conectó la BD con la página web mediante el lenguaje de programación del lado del servidor PHP- *Hypertext Preprocessor* (http://www.php.net/) y también se lo usó para poder acceder a la base de datos, mediante el uso de PHPmyadmin. Esta última es una herramienta escrita en PHP que permite manejar la administración de MySQL a través de páginas web, ayuda a crear y eliminar bases de datos; crear, eliminar y alterar tablas; borrar, editar campos y ejecutar sentencias SQL. Para ello, se entró a https://localhost/dashboard/phpmyadmin/ se seleccionó la opción “Base de datos”, “crear nueva base de datos”, se la nombró “**Eureca**” y se colocó el

cotejamiento **latin1_spanish_ci,** lo cual determina los tipos de caracteres que se puede usar. A pesar de que la BD se escribió en inglés, se decidió usar este cotejamiento porque permite ingresar caracteres especiales como ñ o tildes, lo cual se necesitará para los apellidos de algunos autores.

Luego, se cargaron los datos que se encontraban en Excel (los cuales fueron guardados por columnas, en formato .csv y sin encabezados), para lo cual primero se pasó el diseño de las tablas normalizadas (apartado 2) a la nueva base creada en phpmyadmin. Usando la opción “crear nueva tabla” de la pestaña “estructura”, se colocó el nombre de cada una de las tablas y el número de campos en cada una de ellas. Se colocaron los nombres que identificaban a cada campo y se especificó si es clave primaria o no. Luego se identificó a cada uno de los tipos de datos que son interpretados por el programa. Por ejemplo, el tipo de dato CHAR(n) indica que el dato es de n caracteres (ni más ni menos), VARCHAR(20) indica que el dato puede tener hasta 20 caracteres (aunque el valor máximo sea hasta 255 caracteres), TEXT se usa para datos con más de 255 caracteres (pero algunas veces toma solo hasta 1024 caracteres), BLOB es para guardar archivos en formatos .pdf, .docx e imágenes en binario, FLOAT(s,d) se usa para decimales donde “s=size” especifica los dígitos totales y “d=decimal” el número de dígitos luego del punto decimal, INT(n) es un dato entero cuyo número de dígitos se expresa en el parámetro size=n. Es muy importante esta consideración, ya que si no le otorgamos el “tipo de dato” correcto podemos generar errores en nuestra base datos, sobre todo al hacer consultas. Cada tipo de dato fue identificado según se describe en el manual W3Schools (https://www.w3schools.com/sql/sql_datatypes.asp). El trabajo se guardó usando el motor de almacenamiento InnoDB debido a que maneja mejor las claves foráneas.

3.9. Construcción y validación de un formulario

Las bases de datos biológicas deben ser actualizadas continuamente, por lo que es necesario un formulario de contacto para recibir sugerencias por parte de los usuarios de EURECA o para permitir la adición de datos. Para ello

se confeccionó un formulario y se lo validó (*back-end*) mediante PHP, usando el método POST, por ser más seguro que el método GET.

3.10. Conexión de la base de datos y consultas (pruebas del sistema)

Se conectó la BD con la página web y se programó las consultas que se realizarán en la BD insertando código SQL en PHP. El buen funcionamiento de la base de datos dentro de EURECA se verificó realizando algunas consultas usando SQL.

3.11. Almacenamiento en el hosting

Se compró un dominio por medio de namecheap (https://www.namecheap.com) y se contrató el hosting gratuito "Free Web Hosting Area" (https://doms.freewha.com/) con lo que nuestra página está disponible en la siguiente URL www.eurecabd.com

La trasferencia de archivos desde la computadora personal al servidor se hizo por medio del programa FTP (File Transfer Protocol) denominado Filezilla (https://filezilla-project.org/), el cual se descargó en la PC y se colocaron datos como IP Address, username, password, FTP hostname, FTP username y FTP password provistos por el hosting al correo personal, en el gestor de sitios de Filezilla.

4. RESULTADOS Y DISCUSIÓN

4.1. Elección de los datos de las tablas

A continuación, se muestran las tablas que fueron pensadas en primera instancia para la base de datos dentro de EURECA, en las cuales se identificaron los atributos, los tipos de datos y los dominios.

A) Tabla proteína: en esta pestaña se muestran los datos relacionados a las proteínas del sistema de reparación indirecta del ADN que se pensó en primera instancia.

Nombre del dato	Tipo de dato	Dominio
Nombre	VARCHAR(15)	Conjunto de entre 1 a 15 caracteres
Otros nombres	VARCHAR(15)	Conjunto de entre 1 a 15 caracteres
Género	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Especie	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Organismo	VARCHAR(30)	Conjunto de entre 1 a 30 caracteres
Secuencia FASTA	TEXT	Conjunto de más de 255 caracteres
Número de aminoácidos	INT(6)	Número entero positivo de hasta 6 dígitos
ID en Uniprot/SwissProt	VARCHAR(15)	Conjunto de entre 1 a 15 caracteres
Datos provenientes de SwissProt	FLOAT(4, 3)	Conjunto de números decimales de 4 dígitos y 3 dígitos luego de la coma.
Función	TEXT	Conjunto de más de 255 caracteres

B) Tabla gen: se muestran los datos de los genes del sistema de reparación indirecta del ADN, con su link respectivo a la base de datos correspondiente.

Nombre del dato	Tipo de dato	Dominio
Nombre	VARCHAR(15)	Conjunto de entre 1 a 15 caracteres
Otros nombres	VARCHAR(15)	Conjunto de entre 1 a 15 caracteres
Especie	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Código en el NCBI	VARCHAR(15)	Conjunto de entre 1 a 20 caracteres
Secuencia FASTA	TEXT	Conjunto de más de 255 caracteres
Función	TEXT	Conjunto de más de 255 caracteres

C) Tabla factores bióticos: se ha reportado previamente que los organismos

mutantes en los genes de reparación indirecta del ADN, reciben una respuesta diferente en comparación a los de genotipo *wild type* (WT) cuando son sometidos a factores bióticos. Esta tabla es útil para que los investigadores conozcan los estudios previos que se han publicado y qué factores bióticos ya se han estudiado. Además, permite planificar futuros trabajos que no hayan sido reportados en el organismo de interés. En esta pestaña, para un determinado factor biótico se podrán buscar diversos datos e incluso el link de la publicación correspondiente.

Nombre del dato	**Tipo de dato**	**Dominio**
Factor	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Mutación reportada	BIT	0 es ausencia y 1 presencia, sería lo que se conoce comúnmente.
Diferencia reportada entre WT y mutante	TEXT	Conjunto de más de 255 caracteres
Link de la publicación	TEXT	Conjunto de más de 255 caracteres
Género	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Especie	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Organismo	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Gen afectado	VARCHAR(10)	Conjunto de entre 1 a 10 caracteres

D) Tabla factores abióticos: los organismos mutantes en los genes de reparación indirecta del ADN, reciben una respuesta diferente en comparación a las del genotipo WT cuando son sometidas a factores abióticos tales como sales, azúcares, metales pesados, radiación UV, etc. En esta pestaña se pueden buscar los siguientes datos para un determinado factor abiótico e incluso el link de la publicación correspondiente.

Nombre del dato	**Tipo de dato**	**Dominio**
Factor	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Gen afectado	VARCHAR(10)	Conjunto de entre 1 a 10 caracteres
Link de la publicación	TEXT	Conjunto de más de 255 caracteres
Género	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Especie	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Nombre común de la especie	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Mutación reportada	BIT	0 es ausencia y 1 presencia, sería lo

		que se conoce comúnmente como dato booleano.

E) Tabla fenotipo: en diversas especies se ha reportado que los organismos pueden presentar un fenotipo en particular en presencia de mutaciones en los genes involucrados en los sistemas de reparación indirecta. Por ejemplo, en plantas se ha reportado la presencia de hojas cloróticas, mayor área de la roseta, semillas abortadas; en levaduras se ha observado muerte precoz, entre otros fenotipos. Por lo tanto, será más fácil buscar información sobre estos fenotipos en la base de datos dentro de EURECA.

Nombre del dato	Tipo de dato	Dominio
Fenotipo	TEXT	Conjunto de más de 255 caracteres
Genes involucrados	VARCHAR(10)	Conjunto de entre 1 a 10 caracteres
Género	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Especie	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Organismo	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Link de las publicaciones	TEXT	Conjunto de más de 255 caracteres

F) Tablas publicaciones: en esta ventana se colocaron las publicaciones por autor con información relevante y el link a las mismas.

Nombre del dato	Tipo de dato	Dominio
Genes involucrados	VARCHAR(10)	Conjunto de entre 1 a 10 caracteres
Género	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Especie	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
Organismo	VARCHAR(20)	Conjunto de entre 1 a 20 caracteres
PubMed ID	VARCHAR(15)	Conjunto de entre 1 a 15 caracteres
Link	TEXT	Conjunto de más de 255 caracteres

Cabe aclarar que las tablas presentadas en esta sección deben ser normalizadas antes de incluirlas en la página web.

4.2. Normalización del modelo relacional

La relación entre una tabla principal y una secundaria se lleva a cabo por medio de las claves primarias y foráneas. La clave primaria (PK) de una tabla

es el único dato en el modelo relacional que provoca redundancia, ya que se duplica en las claves foráneas (FK) de otra tabla, lo que permite establecer las relaciones. En este trabajo, se eligieron las PKs en base a que sus valores son conocidos y distintos de nulo, tienen una codificación sencilla, el contenido de sus valores no varía, la memoria que ocupan es mínima y es lo más relevante de la tabla, ya que identifica de forma única a cada fila de una tabla. Sin embargo, no solo basta con identificar las tablas y las claves primarias y foráneas, sino que además se deben cumplir con las tres formas normales, que según Marqués (2011) son:

1- La primera forma normal (1FN) establece que cada campo debe tener un único valor indivisible (atómico).
2- La segunda forma normal (2FN) establece que se cumpla la primera forma normal y obliga a la creación de tablas adicionales cuando alguna tabla tenga datos que no tengan ninguna relación con la clave primaria.
3- La tercera forma normal (3FN) establece que se cumplan las dos formas anteriores y que ninguna columna, que no pertenezca a la clave primaria, puede depender del valor de otra columna.

Ocasionalmente la tercera forma normal puede degradar en gran medida la respuesta de la base de datos, por lo que, hay situaciones que ameritan que se rompa esta regla.

En el caso de la base de datos dentro de EURECA tenemos seis tablas: proteína, gen, factores bióticos, factores abióticos, fenotipo y publicaciones para las cuales se evaluó el cumplimiento de cada forma normal. El hecho de establecer el tipo de dato y su dominio en el apartado anterior, donde figuran todos los datos que se quieren presentar en la BD, fue muy favorable para identificar el incumplimiento de las formas normales de las tablas planteadas en primera instancia.

***Tabla proteína*:** al examinar esta tabla vemos que no cumple la 1FN, ya que una proteína puede tener más de un nombre adicional, lo que hace que no sea

un atributo monovaluado y simple, por lo que se tuvo que desglosar. Lo mismo pasa con los datos provenientes de SwissProt los cuales incluyen: peso molecular, punto isoeléctrico, índice de inestabilidad, índice alifático y GRAVY. Por otro lado, se decidió que el atributo función quede en la tabla de genes, ya que la función del gen y la proteína es la misma y sería un dato redundante que ocuparía espacio innecesario en la base de datos. Cuando observamos la definición de la 2FN nos damos cuenta que esta tabla no cumple con ella, ya que existen atributos que guardan relación entre ellos y por ende deben constituir una nueva tabla, como por ejemplo: género, especie y organismo. Todos estos atributos dependen de un "id_especie", pero no de un "id-proteína", por lo que agregaremos una nueva tabla llamada especie. Las características de las proteínas dependen de todos los identificadores de las proteínas independientemente de la base de datos, por lo que se creó una nueva tabla de características de las proteínas.

A continuación, se muestran 6 tablas que cumplen las 3 formas normales con nombre y abreviatura en inglés, y para asegurar su cumplimiento describimos el tipo de atributo en cada una de las tablas.

La primera tabla está constituida por todos los identificadores de las proteínas de las bases de datos Uniprot (IDuniprot) y del NCBI (GI), todas ellas dependen solo del código de identificación que se les dio en la BD. Si estas estaban incluidas, por ejemplo, con atributos de las características de las proteínas, se estaría incumpliendo la 3FN ya que también serían dependientes de "IDuniprot" y no solo de "IDprot" (tabla 4.2.1).

Tabla 4.2.1. Atributos de la tabla de proteínas renombrada "Proteins"

Proteins		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Código de identificación de la proteína en la BD	IDprot (PK)	Monovaluado y simple
Código de la proteína en Uniprot	IDuniprot	Monovaluado y simple
Identificación en el NCBI	GI	Monovaluado y simple

Como consecuencia de esto se creó una segunda tabla con las características de las proteínas que dependen de una PK que es el código de identificación de Uniprot (id_prot) (tabla 4.2.2).

Tabla 4.2.2. Atributos de la tabla de características de las proteínas renombrada "ProteinsChar"

ProteinsChar		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Código de la proteína en Uniprot	id_prot (PK)	Monovaluado y simple
Peso molecular	Weight	Monovaluado y simple
Punto isoeléctrico	pI	Monovaluado y simple
Índice de inestabilidad	instindex	Monovaluado y simple
Índice alifático	alipindex	Monovaluado y simple
Promedio general de hidropatía	GRAVY	Monovaluado y simple
Nombre de la proteína	nameprot	Monovaluado y simple
Número de aminoácidos	aanumber	Monovaluado y simple
Secuencia FASTA	fastaprot	Monovaluado y simple

Para aquellas proteínas que pueden tomar diferentes nombres, se decidió separarlos en una tercera tabla y solo incluir tres nombres adicionales. Si la proteína no presenta algún nombre adicional, se colocó como valor NULL, de tal forma que no interfiera en las posteriores consultas SQL (tabla 4.2.3).

Tabla 4.2.3. Atributos de la tabla "otros nombres de proteínas" renombrada "Protname"

Protname		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Nombre 1 de la proteína	name1	Monovaluado y simple
Nombre 2 de la proteína	name2	Monovaluado y simple
Nombre 3 de la proteína	name3	Monovaluado y simple
Código de identificación de la proteína en la BD	IDname (PK)	Monovaluado y simple

Como se mencionó anteriormente, se separó a los organismos en una tabla independiente, ya que el género, especie y nombre común se encuentran en relación con el código de identificación de especie y no con el ID de la proteína (tabla 4.2.4).

Tabla 4.2.4. Atributos de la tabla de especies renombrada "Species"

Species		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Código de identificación de especie	IDsp (PK)	Monovaluado y simple
Nombre común de la especie	namesp	Monovaluado y simple
Género	gender	Monovaluado y simple
Especie	Sp	Monovaluado y simple

Lo mismo ocurre con los sistemas de reparación indirecta del ADN, los cuales deben estar en una tabla aparte y no había sido tomado en cuenta en primera instancia (tabla 4.2.5).

Tabla 4.2.5. Atributos de la tabla de sistemas de reparación indirectos del ADN renombrada "IndSyst"

IndSyst		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Sistema indirectos de reparación de ADN	Type	Monovaluado y simple
Código de identificación del tipo del sistema de reparación indirecto del ADN	IDsys (PK)	Monovaluado y simple

***Tabla genes*:** al examinar esta tabla vemos la redundancia de atributos que se encuentran en la tabla "Species", por lo que no se volvieron a considerar. Se agregó el atributo IDgen como PK (tabla 4.2.6). Sin embargo, se encuentran varios nombres para un mismo gen, por ello, se decidió agregar una nueva tabla que incluyera tres nombres adicionales y al igual que antes, se colocó NULL cuando no existen nombres adicionales (tabla 4.2.7). De igual manera se procedió para la tabla de características del gen, para que cumpla la 2FN y 3FN (tabla 4.2.8). Se verificó que las nuevas tablas cumplieran las tres FN, quedando así:

Tabla 4.2.6. Atributos de la tabla de genes renombrada "Gene"

Gene		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Código de identificación del gen en la BD	IDgen (PK)	Monovaluado y simple
Código del gen en el NCBI	IDNCBI	Monovaluado y simple

Tabla 4.2.7. Atributos de la tabla de otros nombres de genes renombrada "Genname"

Genname		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Nombre 1 del gen	name1	Monovaluado y simple
Nombre 2 del gen	name2	Monovaluado y simple
Nombre 3 del gen	name3	Monovaluado y simple
Código de identificación del gen en la BD	ID_name (PK)	Monovaluado y simple

Tabla 4.2.8. Atributos de la tabla de características de genes renombrada "GenChar"

GenChar		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Código de identificación del gen en la BD	ID_char (PK)	Monovaluado y simple
Nombre del gen	namegen	Monovaluado y simple
Secuencia FASTA del gen	fastagen	Monovaluado y simple
Función del gen	function	Monovaluado y simple

***Tabla factores bióticos*:** No se consideraron los atributos de tablas redundantes antes descritas, y tampoco fue incluido el atributo "link de la publicación", ya que existe una tabla llamada publicación. En cuanto al atributo "mutación reportada", en primera instancia se pensó como un dato booleano de presencia o ausencia de mutación (en sql BIT). Sin embargo, aporta más información si se describe el género, especie y tipo de patógeno. Por otro lado, los datos de factores bióticos está relacionados a la tabla "Gene" e "IndSys", por lo que, fue necesario incluir un identificador del gen como FK, así como un identificador del sistema indirecto de reparación como atributos, para saber

cuál de ellos se ve afectado. Se verificó que la tabla cumpliera las formas normales (tabla 4.2.9).

Tabla 4.2.9. Atributos de la tabla de factores bióticos renombrada "Genname"

BioticF		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Código de identificación del factor biótico	IDpat (PK)	Monovaluado y simple
Género del patógeno	PatGenus	Monovaluado y simple
Especie del patógeno	Patspecie	Monovaluado y simple
Tipo de patógeno	Pattype	Monovaluado y simple

***Tabla factores abióticos*:** para normalizar esta tabla, se sacaron los atributos redundantes cerciorando que se encontraban en otras tablas y se verificó el cumplimiento de las formas normales. Por otro lado, se decidió colocar el atributo "stress" donde se consideró a los factores abióticos reportados (tabla 4.2.10).

Tabla 4.2.10. Atributos de la tabla de factores abióticos renombrada "AbioticF"

AbioticF		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Código de identificación del factor abiótico	IDstress (PK)	Monovaluado y simple
Efecto de factor abiótico	stress	Monovaluado y simple

***Tabla fenotipo*:** del mismo modo que en las tablas anteriores, se sacaron los atributos redundantes, se conservó solo el atributo de identificación que representa la PK y el atributo de fenotipo es un dato que describe el tipo de fenotipo reportado (tabla 4.2.11). Por otro lado, se debió cambiar a la última tabla el atributo de link de las publicaciones por no cumplir la 2FN y la 3FN. A raíz de esto fue necesario agregar FKs adicionales para que las tablas guarden relación y se puedan realizar las consultas SQL.

Tabla 4.2.11. Atributos de la tabla de fenotipo renombrada "Phenotype"

Phenotype		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Código de identificación del fenotipo	IDphenot (PK)	Monovaluado y simple
Presencia de fenotipo alterado	pheno1	Monovaluado y simple

***Tabla publicaciones*:** en esta tabla solo se incluyó un identificador del paper (PK) y el link del mismo, de forma tal que el investigador pueda consultar con mayor detalle. En todos los casos se verificó que se cumplan las formas normales (tabla 4.2.12).

Tabla 4.2.12. Atributos de la tabla de publicaciones renombrada "Papers"

Papers		
Nombre	**Abreviatura en inglés**	**Tipo de atributo**
Código de identificación del paper	IDpaper (PK)	Monovaluado y simple
Link al artículo original	Linkpaper	Monovaluado y simple

De esta manera, las seis tablas iniciales se desglosaron en doce tablas que modificaron la propuesta original. Las tablas agregadas se encuentran en color gris y se subrayó en todas ellas la PK.

Para realizar las relaciones establecí a las tablas "IndSyst" y "Species" como principales, ya que así fueron definidas en el desarrollo de la página web. Esto significa que la gran mayoría de las tablas convergen en estas dos por medio de las FKs. Cabe recalcar que las abreviaturas de las FKs son diferentes para evitar errores al momento de la programación de las consultas en SQL.

Las tablas provenientes de datos de proteínas deben estar conectadas, para ello se estableció a "Proteins" como la principal, por lo que "ProteinsChar" y "Protname" se relacionaron a esta. Sin embargo, los nombres de las proteínas también se relacionan con sus características, por lo que, se unió

“Protname” con “ProteinChar”. Además, la tabla “Proteins” también se relacionó a “Gene”, “IndSyst” y “Species”.

Por otro lado, las tablas provenientes de datos de genes también deben estar conectadas, por lo que se estableció a “Gene” como la principal y a esta se unieron “Genname” y “GenChar”, y estas dos últimas también se unieron entre ellas, ya que hay una relación evidente entre el nombre del gen y sus características. Por último, “Gene” se relacionó con “Species” e “IndSyst”.

La tabla “Phenotype” se unió a la de los factores bióticos y abióticos. Sin embargo, para que haya una relación entre todos los datos se unió tanto “AbioticF” como “BioticF” a “IndSyst”, “Papers” y a “Species”. También se relacionó a la tabla “Papers” con “IndSyst”.

El modelo relacional de EURECA que incluye las PKs, las FKs y las relaciones se muestra en la figura 4.2.1:

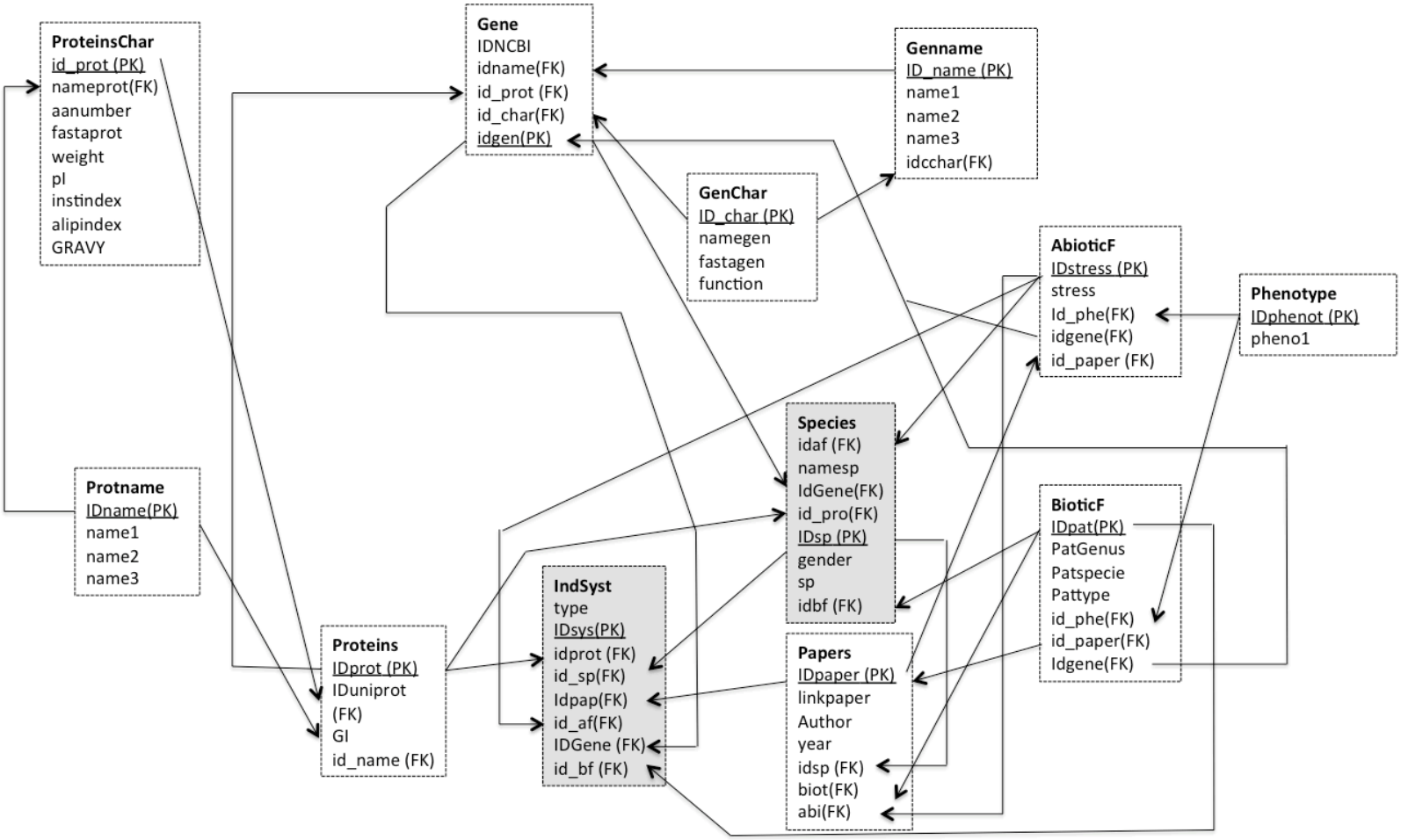


Figura 4.2.1. Modelo relacional de las tablas que conforman EURECA

4.3. Recolección, procesamiento y curación de datos

Los datos de las tablas “Papers”, “Phenotype”, “AbioticF”, “Stress”, “BioticF”, “Pathogen” fueron completados con una extensa revisión bibliográfica de al menos 100 publicaciones científicas. Por otro lado, la tabla “IndSyst” solo tiene como datos los tipos de sistemas indirectos (BER, NER y MMR).

Para las tablas de proteínas, genes y las que derivan de estas, se realizó un análisis bioinformático tal como se describió en el apartado de materiales y métodos. Este análisis fue realizado para cada una de las proteínas de cada uno de los tres sistemas indirectos de reparación del ADN y para cada organismo estudiado. Debido a la gran cantidad de trabajo que este análisis conllevó y considerando el espacio que llevaría mostrar cada uno de ellos (siendo el mismo procedimiento), solo se presenta como ejemplo el análisis bioinformático para el gen y la proteína MSH7 de *A. thaliana*.

En el caso del gen *MSH7* y todos los demás genes, los datos se tomaron del NCBI, colocando en el buscador el nombre del gen y seleccionando la opción “Gene”. Una vez obtenido el *output*, se verificó que corresponda al nombre del gen buscado y al organismo, como se puede ver en la figura 4.3.1. De esta búsqueda se obtuvo el código del gen en el NCBI, los nombres alternativos y la función del gen (figura 4.3.1).

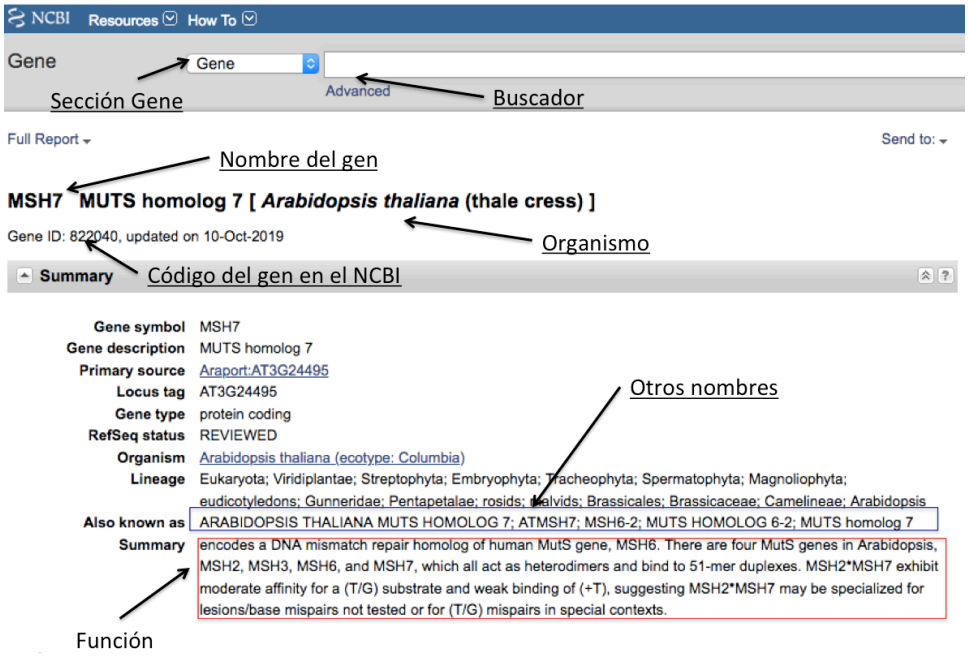


Figura 4.3.1. Resultados de la búsqueda del gen *MSH7* en la base de datos del NCBI

En el mismo *output* de búsqueda de *MSH7*, desde la sección "genomic sequence" se obtuvo la secuencia FASTA haciendo click en el nombre (figura 4.3.2). Puede verse en la figura 4.3.2 que en la URL se aprecia un número que corresponde al ID del gen *MSH7* en la base de datos del NCBI (822040), descrito en la figura anterior. También se observa un nuevo código que corresponde a la secuencia del gen *MSH7* (NC_003074.8).

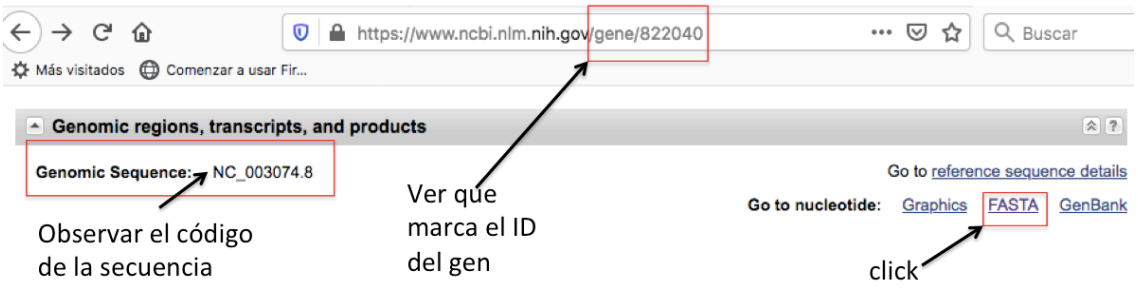


Figura 4.3.2. Verificación de datos para obtener la secuencia FASTA

De esta forma se obtuvo la secuencia del gen en formato fasta (figura 4.3.3), la cual se copió en la BD. Nuevamente en esta figura puede verse que

la URL contiene un nuevo código que corresponde al de la secuencia del gen, lo cual permite verificar que se están extrayendo los datos correspondientes al gen de interés.

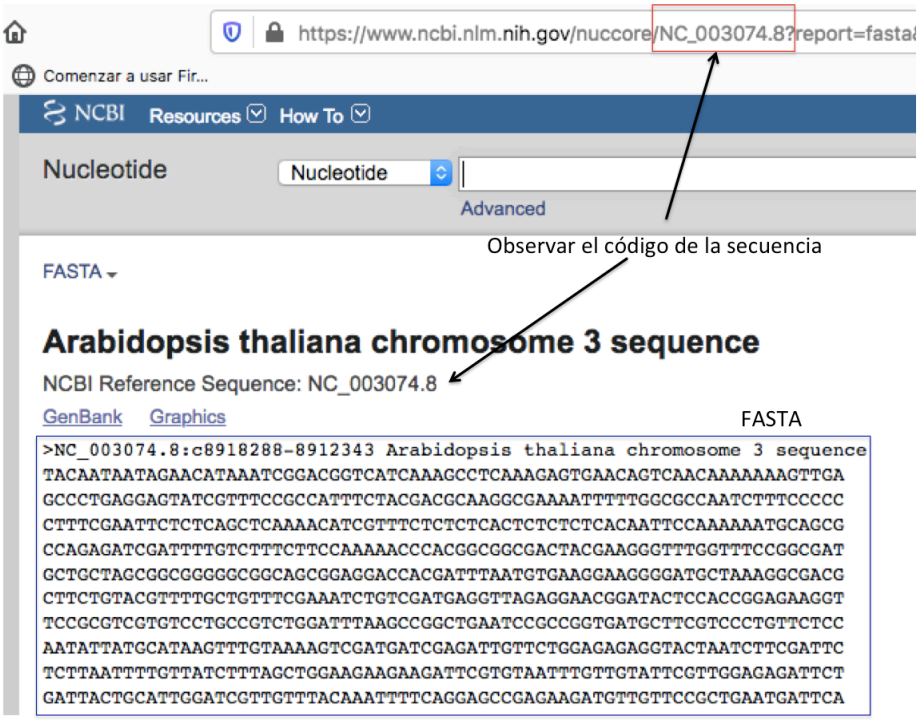


Figura 4.3.3. Secuencia FASTA del gen *MSH7*

En la figura 4.3.3 se muestran las características del formato FASTA (rectángulo azul), el cual comienza con una descripción en una única línea (línea de cabecera), seguida por líneas de datos de secuencia. La línea de descripción se distingue de los datos de secuencia por el símbolo “>” (mayor que) en la primera columna. La palabra siguiente a este símbolo es el identificador o código de la secuencia, y el resto de la línea es la descripción.

Para los datos de proteínas se usó la base de datos UniProt, para lo cual se ingresó en el buscador el nombre de cada una de ellas y se obtuvo el *output* que se muestra como ejemplo en la figura 4.3.4. De esta información se puede obtener el código de identificación de la proteína en esta BD (rectángulo azul), así como su función (rectángulo rojo); sin embargo, esta última información es redundante con la obtenida en el NCBI, por lo que no se consideró en la tabla de proteínas.

Figura 4.3.4. *Output* de la búsqueda de MSH7 en UniProt

En el *output* anterior se obtiene una sección denominada “Sequence” (figura 4.3.5), donde se especifica el largo de la proteína, en este caso 1109 aminoácidos y la masa molecular de la proteína (rectángulo azul), en este caso 122270 Da. Asimismo, se verifica el código de identificación de la proteína (círculo rojo) y a la izquierda se observa su secuencia FASTA junto con un botón que permite su descarga (rectángulo verde).

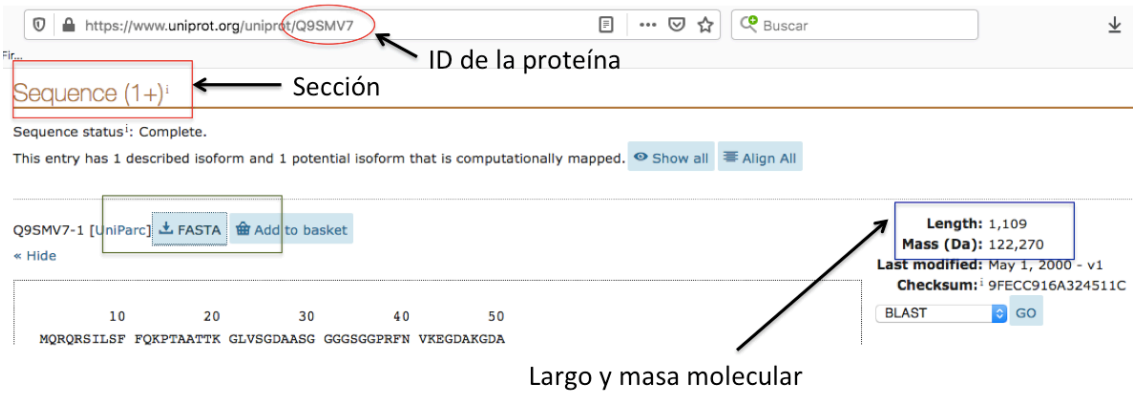


Figura 4.3.5. Continuación del *output* de la búsqueda de MSH7 en UniProt

Para obtener los otros parámetros de la proteína, se usó la herramienta “ProtParam” del Expasy (figura 4.3.6), donde se colocó la secuencia FASTA obtenida en el paso anterior.

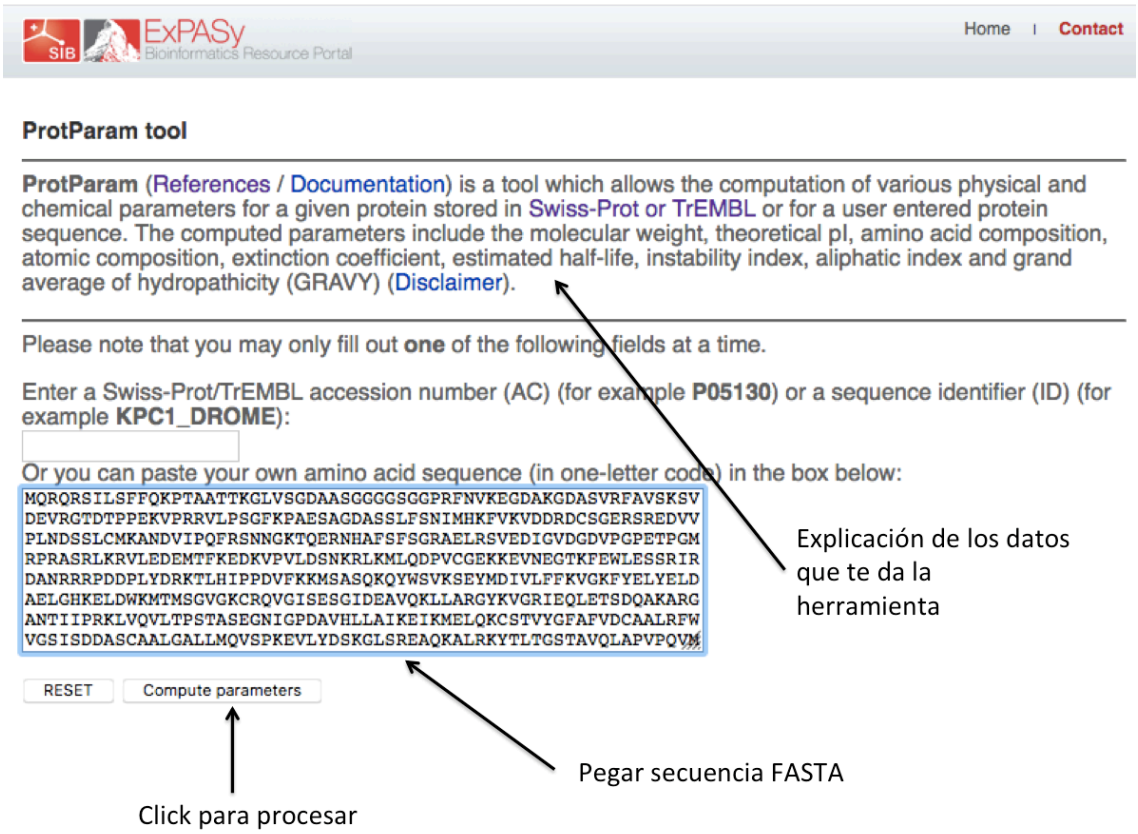


Figura 4.3.6. *Input* de ProtParam

En el *output* de ProtParam (figura 4.3.7), se obtienen diversos datos como el número total de aminoácidos, la composición de aminoácidos, la composición atómica, la fórmula química, el número total de átomos, el coeficiente de extinción, la vida media estimada y aquellos datos que son de interés para la confección de la BD, que incluyen peso molecular, punto isoeléctrico teórico (rectángulo rojo), índice de inestabilidad (rectángulo azul), índice alifático (rectángulo verde) y GRAVY (rectángulo anaranjado).

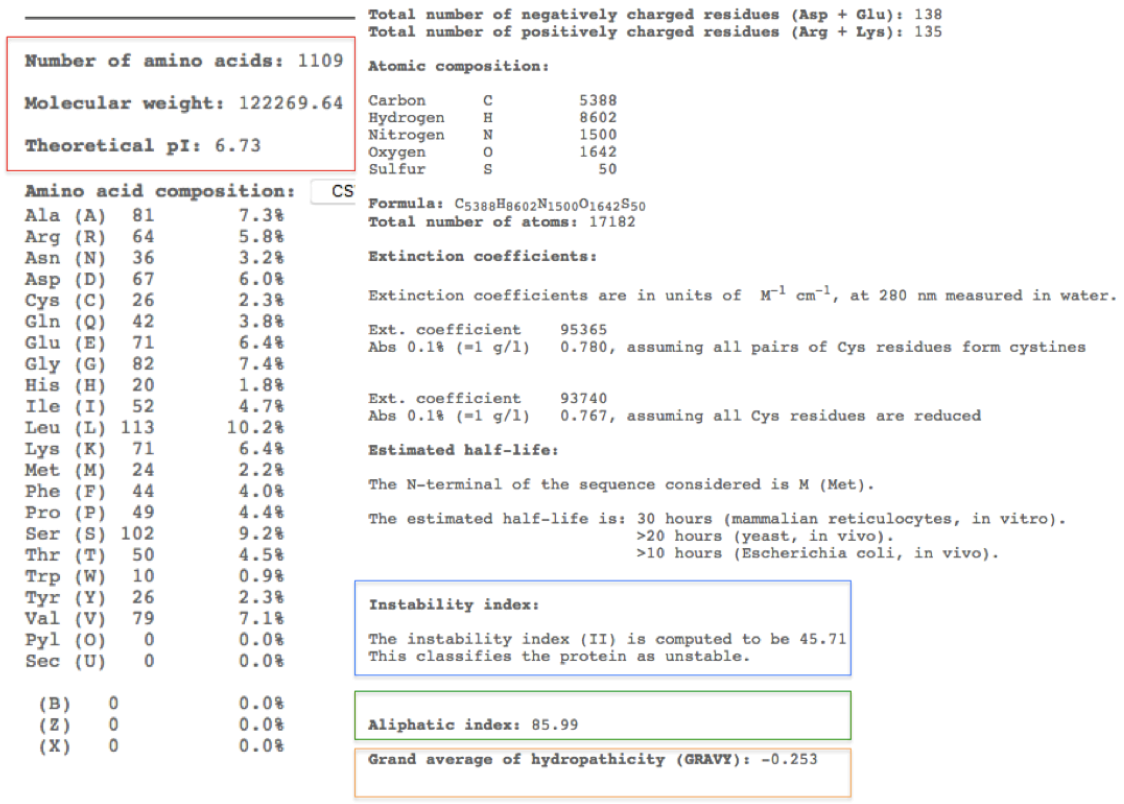


Figura 4.3.7. *Output* de ProtParam

De esta forma se obtuvieron todos los datos para cada una de las proteínas y genes y se guardó la información en una planilla de datos usando Excel. Los demás datos fueron recopilados a partir de artículos científicos, tal como se describió en el apartado de materiales y métodos.

4.4. Esquematización de la página web

El primer paso para el diseño de una página web es definir la arquitectura de la misma, es decir, las páginas que tendrán el sitio web, la organización de las pestañas y donde nos direccionarán. Esta arquitectura de información es también llamada diseño de información y se busca establecer una estructura del sitio web que permita organizar el contenido lógicamente y que facilite la localización de la información. La esquematización de la página web puede ser responsable de la funcionalidad de la búsqueda, los diagramas del sitio y de cómo el contenido y los datos se organizan en el servidor. La arquitectura de la

información está relacionada con el diseño de UX (experiencia usuario) y UI (interfaz usuario) (Förster, 2008; Niederst Robbins, 2012).

La figura 4.4.1 muestra el árbol de contenidos, similar a un mapa conceptual con la información de EURECA.

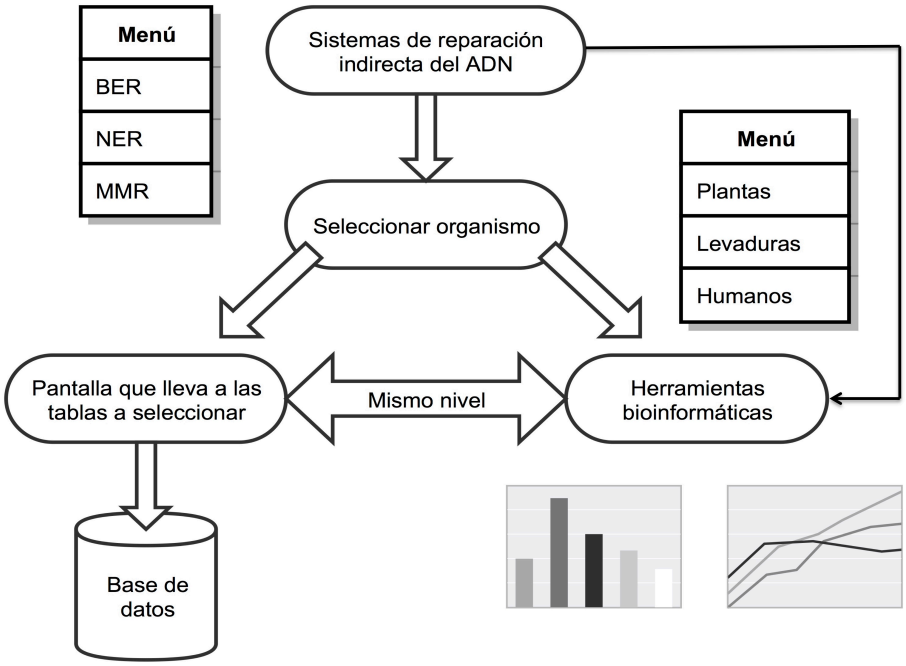


Figura 4.4.1. Arquitectura de la información de EURECA

En la página de inicio, se muestra la información concerniente a los sistemas de reparación indirecta del ADN (figura 4.4.1), donde se debe elegir el sistema de interés a través de las pestañas BER, NER o MMR. Una vez elegida la opción, se re-direccionará a la página que contiene al organismo en estudio, ya sean plantas, levaduras o humanos. Al seleccionarlo, se mostrarán las tablas que contienen los datos curados y la pestaña de herramientas bioinformáticas. Al hacer click en alguna de las opciones de tablas, se mostrará la base de datos de genes, proteínas, factores bióticos, factores abióticos, fenotipo y publicaciones para poder realizar consultas. Por otro lado, si se selecciona la pestaña de herramientas bioinformáticas, se tendrá acceso a las ventanas de búsqueda de motivos microsatélites, análisis de sitios de

restricción y análisis de vector en fase. Debido a que las herramientas bioinformáticas son útiles para todos los casos, independientemente del tipo de sistema de reparación indirecto u organismo que se estudie, se decidió colocar un enlace a estas herramientas en la página de inicio (flecha negra de la figura 4.4.1).

A continuación, se muestra una representación gráfica de algunas pestañas y la organización y contenido dentro de ellas (figura 4.4.2). Por ejemplo, la página de inicio (*index*) contiene una descripción del contenido del sitio web y también los tres mecanismos de reparación indirecta del ADN. Al ingresar a la pestaña específica de cada uno de los sistemas de reparación indirecta del ADN el usuario encuentra una breve descripción de ellos.

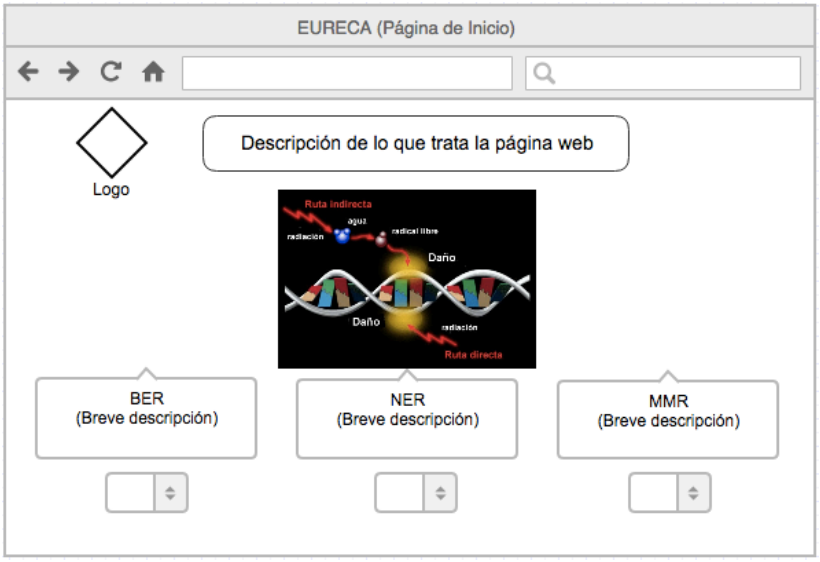


Figura 4.4.2. Esquema de la página inicio que contiene a las pestañas que corresponden a los sistemas de reparación indirectos del ADN que se deben seleccionar (BER, NER y MMR).

En la segunda página es posible seleccionar el organismo a estudiar, con una breve descripción del mismo. Lo que incluye imágenes que representan a los organismos con sus respectivos nombres, como podemos observar en la figura 4.4.3.

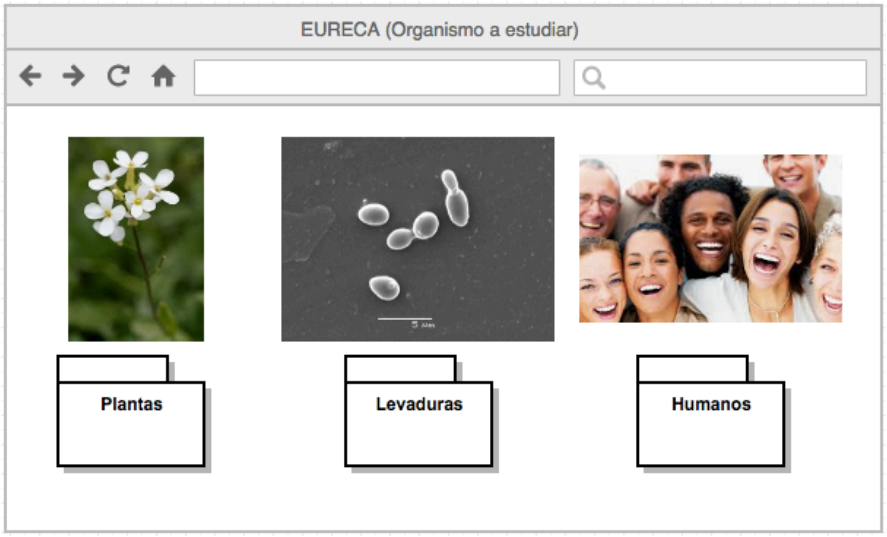


Figura 4.4.3. Esquema de la página con los organismos a estudiar

La tercera página es una pantalla específica que se divide en dos partes, la primera donde están los datos de las tablas y la segunda donde están las herramientas bioinformáticas. Cada sección con su respectiva descripción, para orientar y facilitar la navegación del usuario (figura 4.4.4).

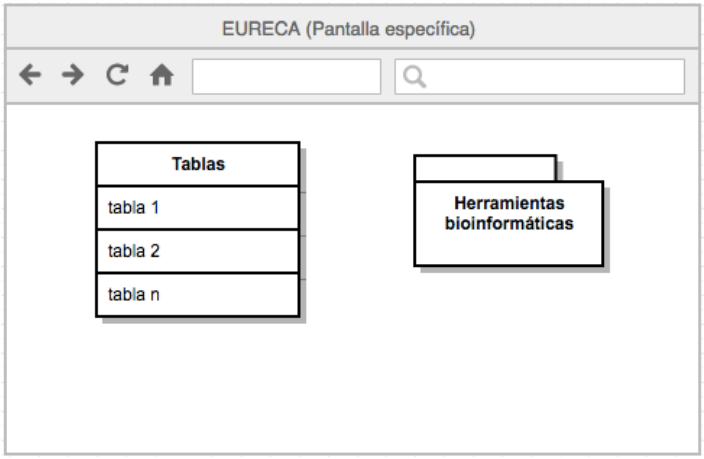


Figura 4.4.4. Esquema de la pantalla específica para seleccionar acceso a la BD y consultas o a las herramientas bioinformáticas

La ventana de herramientas bioinformáticas, tiene 3 pestañas que corresponden a la búsqueda de motivos microsatélites, secuencia en fase y análisis de sitios de restricción (figura 4.4.5), cada uno de ellos con su respectiva descripción.

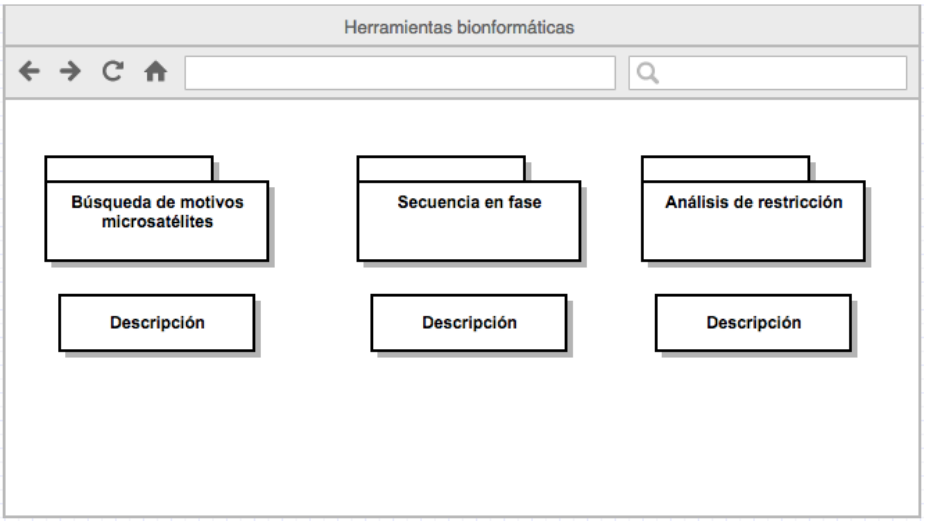


Figura 4.4.5. Esquema de la pantalla "herramientas bioinformáticas"

En la pestaña "búsquedas de motivos microsatélites", se colocaron tres *inputs*: i) donde se especifica si se desea buscar dímeros (ej: AG), trímeros (ej: AGC), tetrámeros (ej: AGCT), etc; ii) donde se especifica el número de repeticiones, por ejemplo si se coloca "3" para el dímero "AG", significaría que AG se repite 3 veces, es decir, "AGAGAG"; iii) campo para colocar la secuencia de ADN. Por último, se incluyeron dos opciones: una para limpiar las ventanas donde se colocan los *inputs* y la otra para procesar la información y encontrar los motivos microsatélites o SSR (figura 4.4.6).

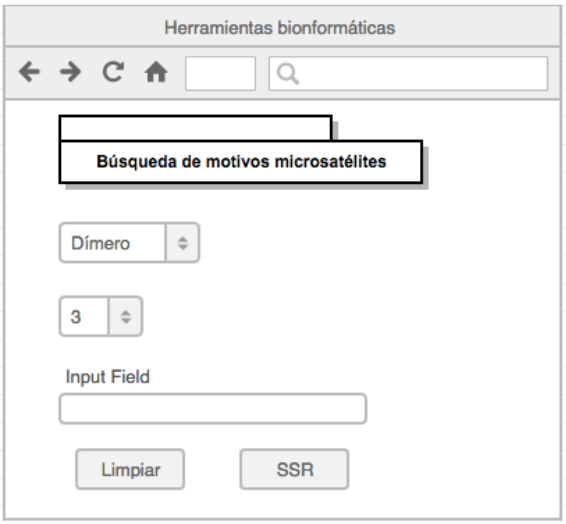


Figura 4.4.6. Pestaña específica con los *inputs* de la búsqueda de motivos microsatélites.

La pestaña “secuencia en fase” solo contiene un *input* para colocar la secuencia de ADN, luego se tiene la opción de limpiar la pestaña o procesar los datos (figura 4.4.7).

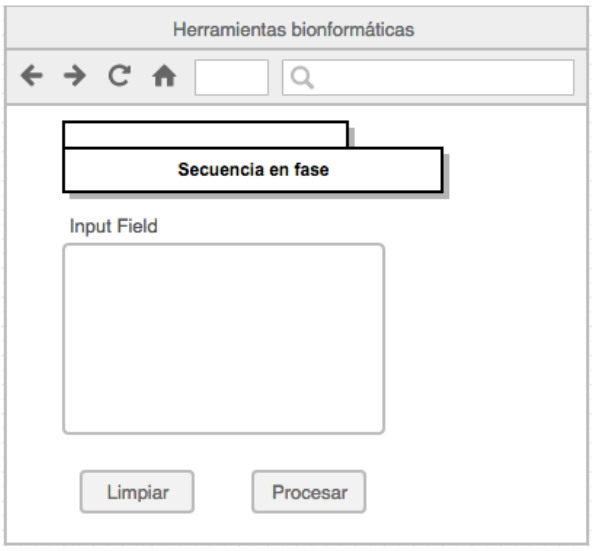


Figura 4.4.7. Pestaña específica de secuencia en fase con su respectivo *input*.

La pestaña de análisis de enzimas de restricción, solo tiene un *input* para colocar la secuencia de ADN y conocer las tijeras moleculares que pueden cortar la secuencia blanco (figura 4.4.8).

Figura 4.4.8. Pestaña específica del análisis de enzimas de restricción con su respectivo *input*.

Es importante mencionar que habrá un link de contacto. Este formulario servirá para recibir sugerencias de los usuarios, así como para recibir su ayuda en la actualización de EURECA. La estructura del formulario es simple, tal como se muestra en la figura 4.4.9.

Figura 4.4.9. Esquema de la página de contacto

Si bien las figuras de la estructura de la página web están en español, esto es solo para ejemplificar, ya que la página web se redactó en inglés.

4.5. Diseño y programación de la página web

En este trabajo se usó Adobe Illustrator para los dibujos de la página web, porque su manejo es mucho más fácil, sobre todo para el tipo de estructuras que se dibujaron y que son originales para esta tesina. Por ejemplo, el logo de Eureca tiene una planta cuyo tallo representa a una molécula de ADN que posee una base dañada, este ADN también pertenece a la levadura, la cual engloba parcialmente a la planta, y la letra “A” de Eureca está representada por un alicate que es una herramienta muy conocida y en este caso se usa como analogía para la reparación del ADN (figura 4.5.1).

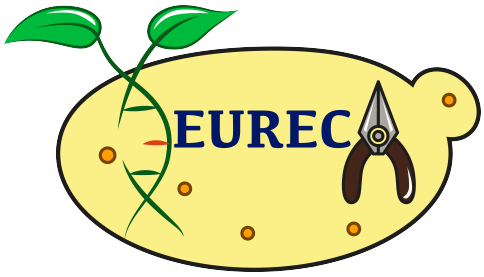


Figura 4.5.1. Logo de EURECA

A continuación, se muestran los dibujos que se diseñaron y vectorizaron en Illustrator para los tres sistemas de reparación indirectos: A) BER, representado por una secuencia de ADN con la base citosina dañada junto a la enzima ADN glicosilasa encargada de su eliminación; B) NER, representado por un sol que transmite radiación UV responsable de la formación de dímeros de pirimidina; C) MMR, representado por una secuencia de ADN que contiene un apareamiento incorrecto (A-G) y las proteínas del sistema MMR que participan en su reconocimiento y reparación (figura 4.5.2).

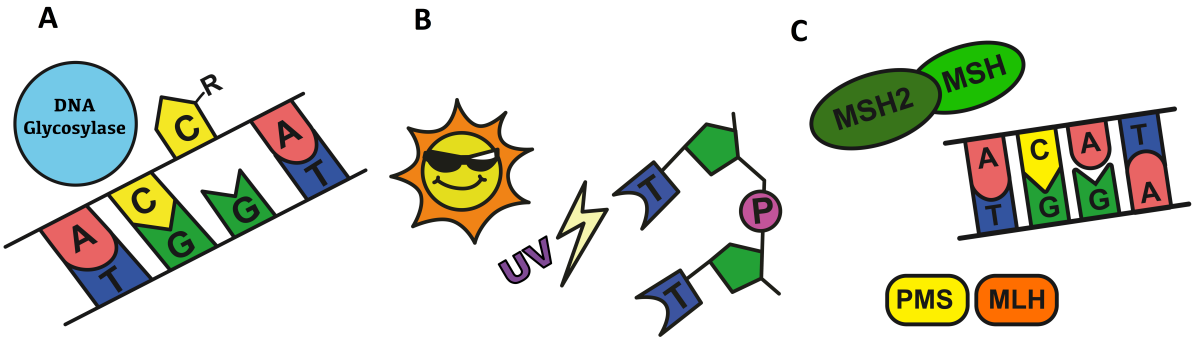


Figura 4.5.2. A) BER, B) NER y C) MMR

Por otro lado, se representaron con dibujos a los organismos incluidos en la BD: plantas (figura 4.5.3A), levaduras (figura 4.5.3B) y humanos (figura 4.5.3C).

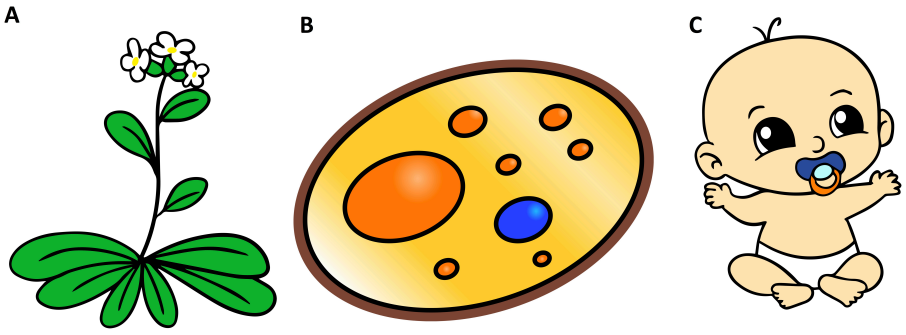


Figura 4.5.3. A) Planta, B) Levadura y C) Humano

Para la pestaña de herramientas bioinformáticas, se representó una molécula de ADN donde los números usados en el código binario (0 y 1) sustituyen a los puentes de hidrógeno. Esto se encuentra dentro de paréntesis y terminado con punto y coma, representando el código de programación y la palabra *Bioinformatics* entre los signos de mayor y menor, para representar el lenguaje de marcado usado para el desarrollo de la página web (figura 4.5.4).

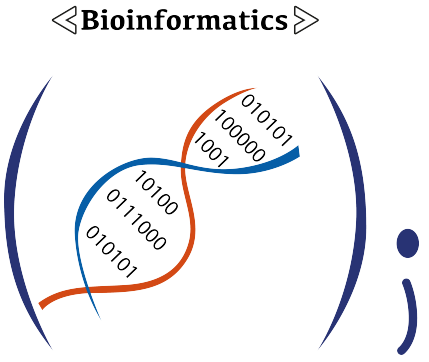


Figura 4.5.4. Dibujo que representa bioinformática

Se conoce como desarrollo web *Front-end* a todo lo que está a la vista e interactúa con el usuario o cliente. Para esto se emplean lenguajes como HTML, CSS y JavaScript (Lathrop, 1999; Luján Mora, 2002; Gauchat, 2012). Una vez desarrollada la página web, es altamente recomendable validar algunos de los datos ingresados en la máquina cliente antes de enviarlos al servidor (Förster, 2008).

Para el desarrollo de esta página web, se usó el lenguaje de marcado HTML. Para establecer el estilo, color, forma y diseño del encabezado, se

utilizó un archivo CSS (encabezado.css) enlazado dentro del documento HTML con el atributo href (un atributo define detalles de comportamiento o presentación de la etiqueta). El nombre y logo de EURECA, se colocaron a manera de *header* (en la parte superior) mediante una imagen incrustada usando <img src="imágenes/logonom.png">. El archivo principal, del cual las demás pestañas parten es "index.html" que en este trabajo es igual a "inicio.html".

El archivo "index.html" fue usado como base para las pestañas "BER", "NER" y "MMR". Adicionalmente, dentro del documento HTML se incluyó un código que permite dirigirse a las pestañas de planta, humano o levadura, al darle *click* a la figura respectiva. Lo mismo ocurre con "bioinfo.html" que es la pestaña de las herramientas bioinformáticas, solo que aquí se enlaza (por medio de href) a imágenes enlazadas a las pestañas de búsqueda de microsatélites (microsatelites.html), secuencia en fase (fase.html) y análisis de sitios de restricción (restricción.html). La figura 4.5.5 ejemplifica la lógica de los archivos, los cuales parten de "index.html". El código HTML completo y comentado de la página web, se puede encontrar en el **anexo 6.1.**

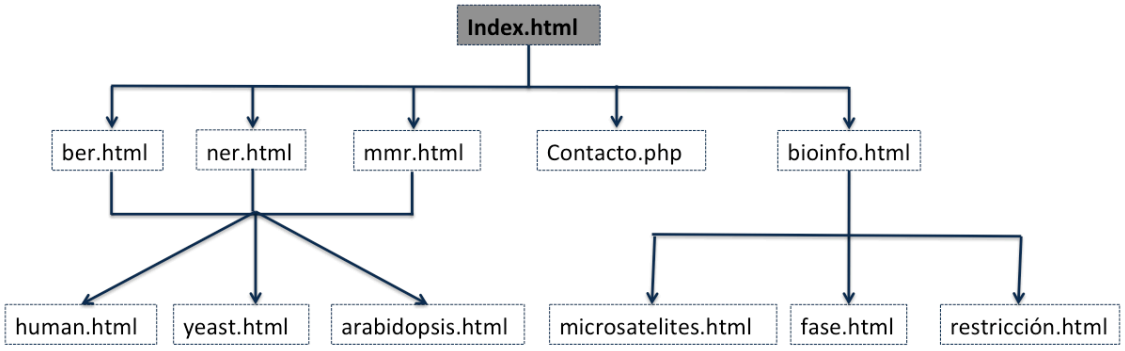


Figura 4.5.5. Lógica de los archivos que conforman la página web de EURECA

En cuanto al estilo, la estructura básica de CSS se basa en reglas que tienen el siguiente formato: **selector** que es el elemento HTML que vamos a seleccionar del documento para aplicarle un estilo concreto, **propiedad** que es una de las diferentes características que brinda el lenguaje CSS y **valor** donde cada propiedad CSS tiene una serie de valores concretos, con los que tendrá

uno u otro comportamiento (W3Schools; Gauchat, 2012). El estilo CSS usado emplea el modelo de cajas, ya que una página web es en realidad un grupo de cajas ordenadas siguiendo ciertas reglas. Las medidas en CSS se emplean, entre otras, para definir la altura, el ancho, los márgenes de los elementos y para establecer el tamaño de letra del texto (Gauchat, 2012). Todas las medidas, se indican como un valor numérico entero o decimal seguido de una unidad de medida. Estas medidas pueden ser absolutas y relativas. En este caso, se incluyen unidades relativas, porque son las que con mayor flexibilidad se adaptan a los diferentes medios (*responsive*). Las medidas relativas definen su valor en relación con otra medida, por lo que, para obtener su valor real, se debe realizar alguna operación con el valor indicado.

Por otro lado, la declaración en CSS indica *"lo hay que hacer"* y el selector indica *"a qué parte"*. El código CSS usado para la página web, para la normalización y para hacerla *responsive*, así como su explicación, se encuentra en el **anexo 6.2.**

Cuando se ejecuta el código HTML en el navegador, se puede apreciar como se verá en la web. Como se mencionó el encabezado es fijo y todas las pestañas lo tienen (figura 4.5.6). El navegador también es el mismo (figura 4.5.6 en letras verdes).

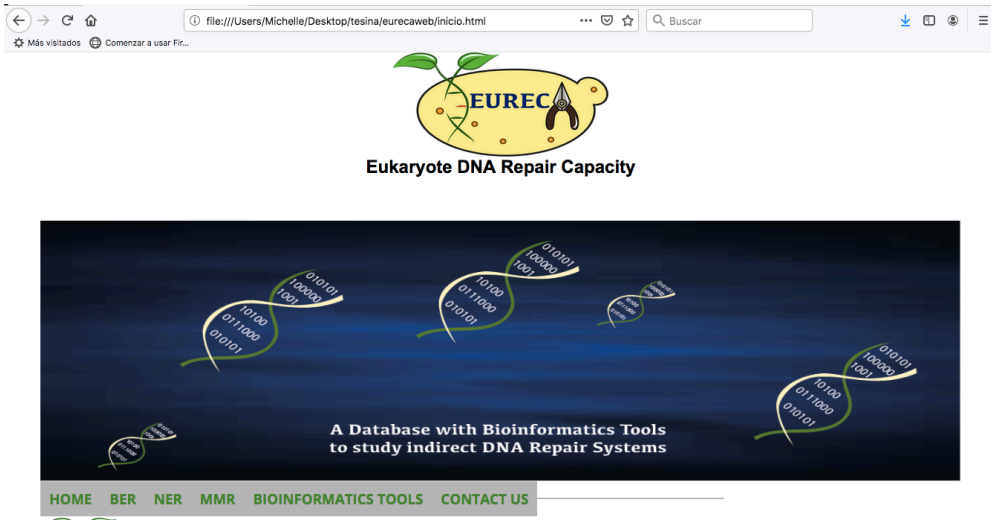


Figura 4.5.6. Se muestra el encabezado de todas las pestañas de la web

Cabe recalcar que en este trabajo los archivos *index* e inicio son lo mismo y se ven en el navegador como en la figura 4.5.7.

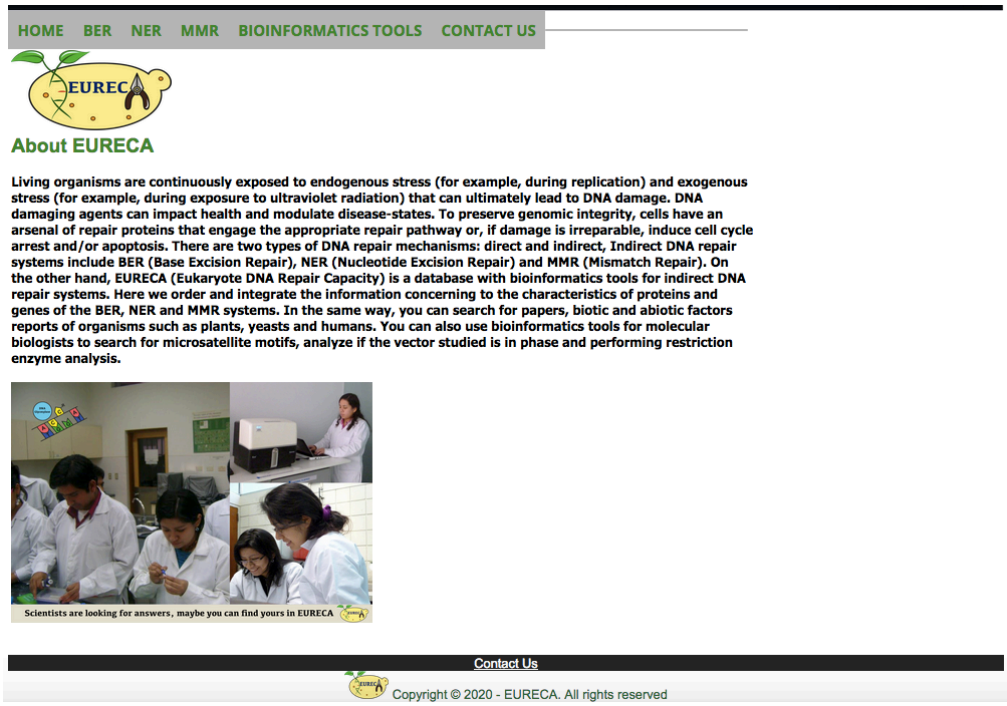


Figura 4.5.7. Archivo "index.html" ejecutado en el navegador.

Por su parte, la ejecución en el navegador de la pestaña correspondiente a los sistemas de reparación BER, NER y MMR, resulta en las figuras 4.5.8, 4.5.9 y 4.5.10, respectivamente.

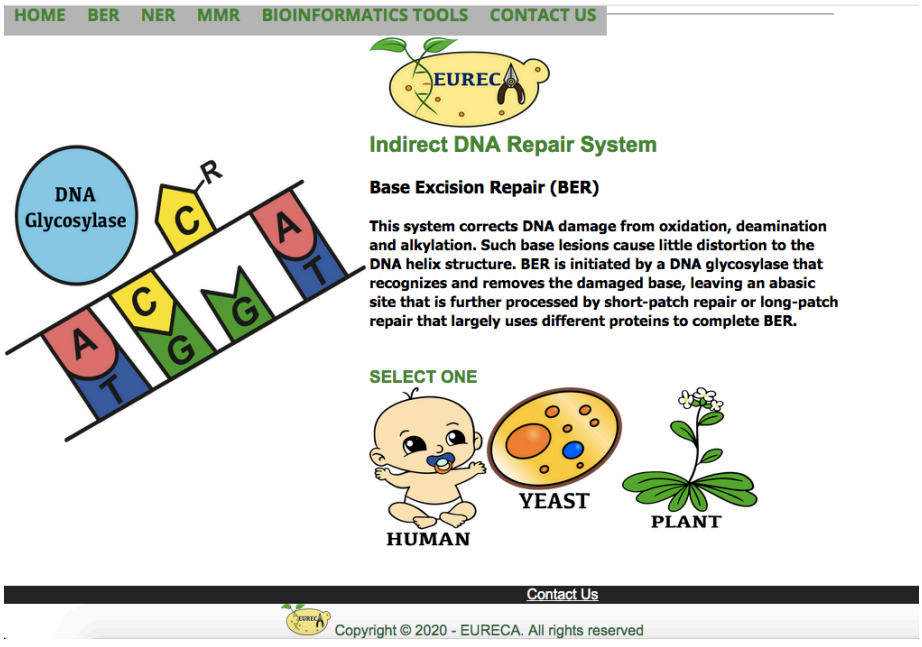


Figura 4.5.8. Archivo "ber.html" ejecutado en el navegador.

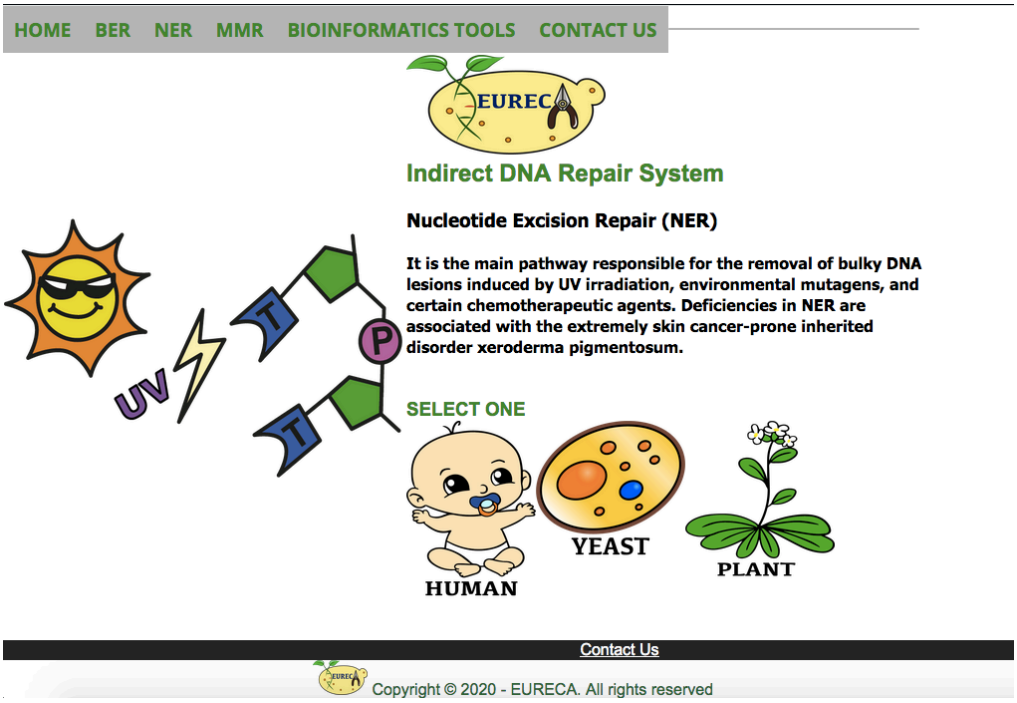


Figura 4.5.9. Archivo "ner.html" ejecutado en el navegador.

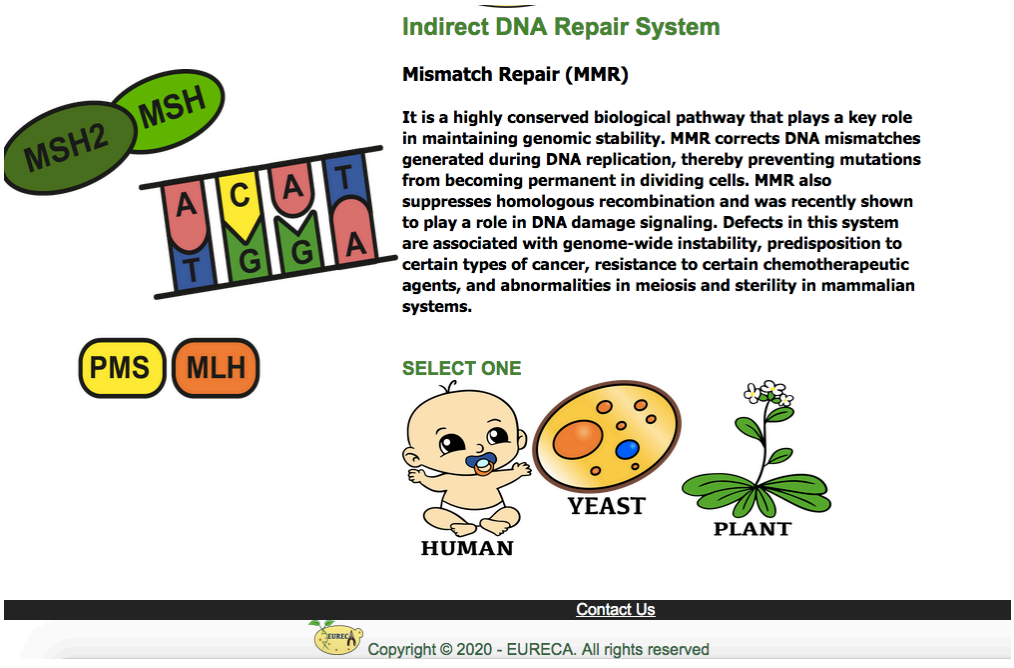


Figura 4.5.10. Archivo "mmr.html" ejecutado en el navegador.

Cada vez que se haga *click* en una pestaña de algún sistema de reparación aparecen las tablas y se pueden realizar consultas. Una vez que el usuario elija un sistema de reparación, tiene la opción de elegir un organismo como planta, humano o levadura. De la figura 4.5.11 a la 4.5.13 se muestran

cómo se ven los archivos correspondientes a especies una vez ejecutados en el navegador.

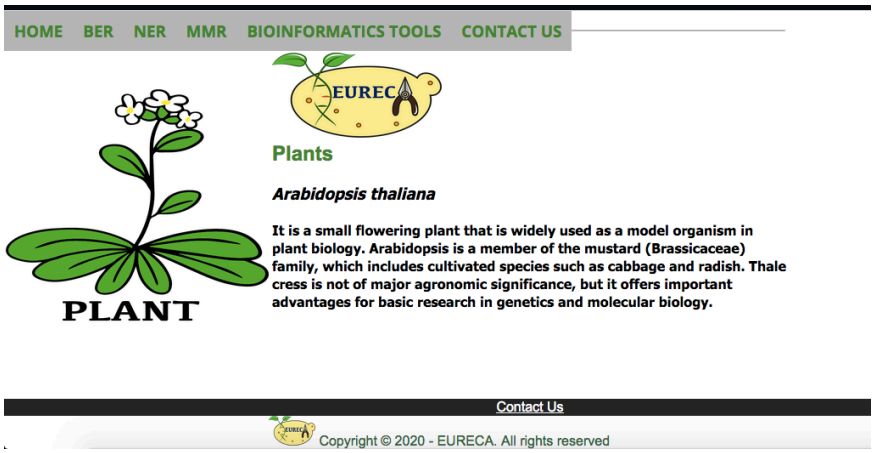


Figura 4.5.11. Archivo "arabidopsis.html" ejecutado en el navegador.

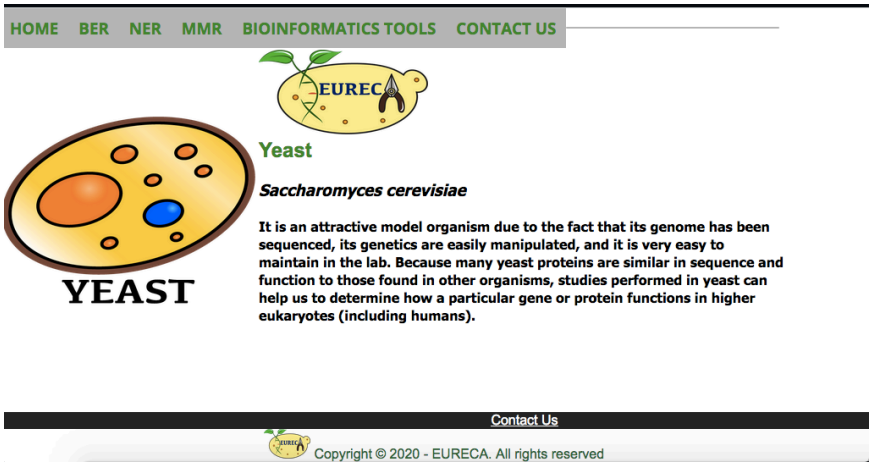


Figura 4.5.12. Archivo "yeast.html" ejecutado en el navegador.

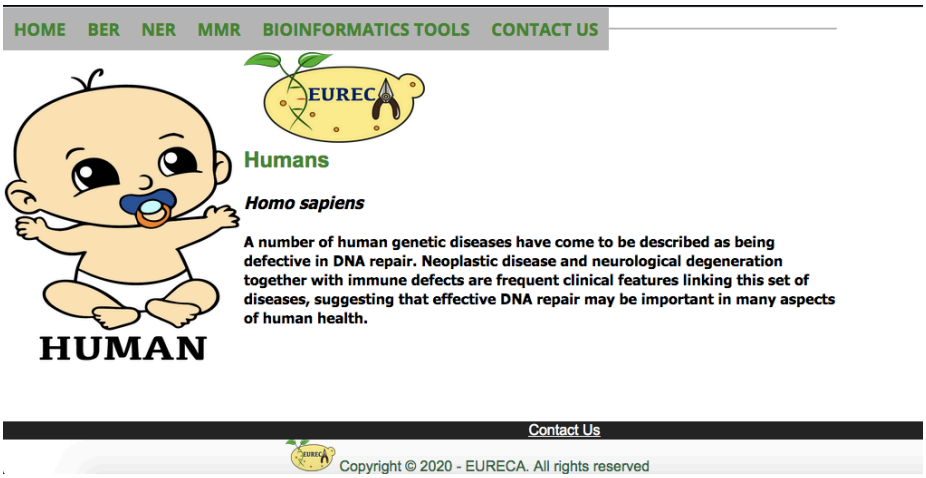


Figura 4.5.13. Archivo "human.html" ejecutado en el navegador.

Por otro lado, la pestaña de herramientas bioinformáticas, se incluye en el navegador (nav) del archivo “index.html”, por lo que siempre habrá acceso a estos *scripts* automáticos de información bioinformática. Una vez en esta pestaña se puede elegir entre las tres herramientas a usar. La pestaña correspondiente a herramientas bioinformáticas (bioinfo.html) ejecutada en el navegador se muestra en la figura 4.5.14.

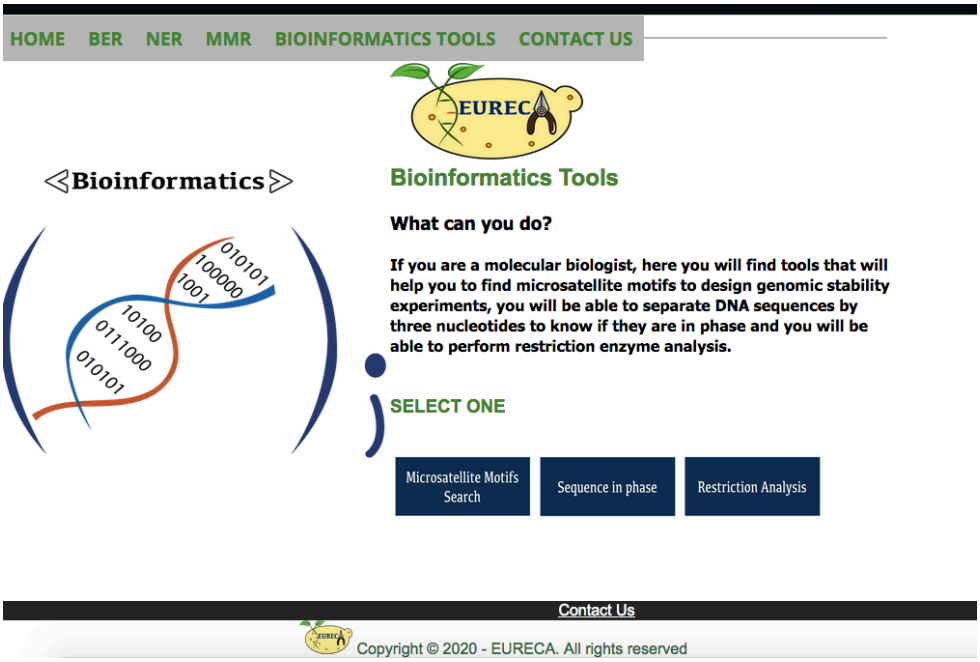


Figura 4.5.14. Archivo “bioinfo.html” ejecutado en el navegador.

Para todo el desarrollo de la página web se usó el editor de texto NotePad++ que fue diseñado específicamente para editar el código fuente de programas informáticos. Tiene la ventaja de simplificar y acelerar la escritura de código fuente resaltando la sintaxis, autocompletando y en algunos casos a medida que el programador escribe, avisa si encuentra algún tipo de problema, permitiendo de esta forma detectar más fácilmente los errores.

4.6. Programación e implementación de las herramientas bioinformáticas

A) *Búsqueda de motivos microsatélites*.- los microsatélites son secuencias de ADN en las que un fragmento se repite de manera consecutiva, generalmente se encuentran en zonas no codificantes del ADN y poseen una alta tasa de mutación (Sun y col., 2012; Chirinos-Arias y col., 2015). Algunos

de ellos pueden encontrarse en el ADN regulador o incluso en el codificante, por lo que, sus mutaciones (variaciones en el número de repeticiones) pueden conducir a cambios fenotípicos. De aquí que el mantenimiento de la estabilidad genómica sea importante y reparado por mecanismos como el sistema MMR que reconoce inserciones y deleciones en el ADN (Spampinato y col., 2009).

Cuando estos mecanismos fallan en reparar el error en el ADN, se produce la inestabilidad de microsatélites (MSI por sus siglas en inglés). Uno de los casos más conocidos de MSI se reporta en el cáncer colorrectal. En los pacientes con cáncer, es necesario la identificación de motivos microsatélites en una secuencia de ADN proveniente de tejido no tumoral y compararla con el aumento o disminución del número de repeticiones del motivo microsatélite del tejido tumoral, lo que ayuda a establecer un pronóstico.

Para desarrollar esta herramienta bioinformática, se creó el archivo “microsatelites.html” que se muestra al cliente y que está enlazado a “bioinfo.html” del navegador y a “analisisdna.js”, el cual contiene el código fuente para encontrar los motivos microsatélites. Los códigos HTML, CSS y JavaScript completos y comentados para desarrollar esta herramienta, se encuentran en el **anexo 6.3.**

La herramienta bioinformática funciona, cuando se ingresa en el archivo “microsatelites.html” la secuencia de ADN (incluso una en formato FASTA), se selecciona el motivo microsatélite de dos a diez nucleótidos y se ingresa el número de repeticiones que se busca. El archivo “análisis.js” valida que la cadena ingresada esté compuesta solo por las letras A, G, C y T que corresponden a las bases nitrogenadas del ADN. Si se coloca un carácter diferente, el *script* se termina. Si se coloca una cadena válida, el *script* crea y busca el motivo microsatélite del largo solicitado. Luego, busca que la siguiente cadena sea igual al motivo especificado inicialmente, si no lo es, elige el motivo siguiente en la cadena de ADN y realiza la misma lógica. Por otro lado, si el segundo motivo es idéntico al primero, busca que se cumpla que el largo (motivo por número de repeticiones) coincida, si lo hace se muestra el resultado al usuario. Caso contrario busca otro motivo hasta terminar la cadena de ADN asignada (figura 4.6.1).

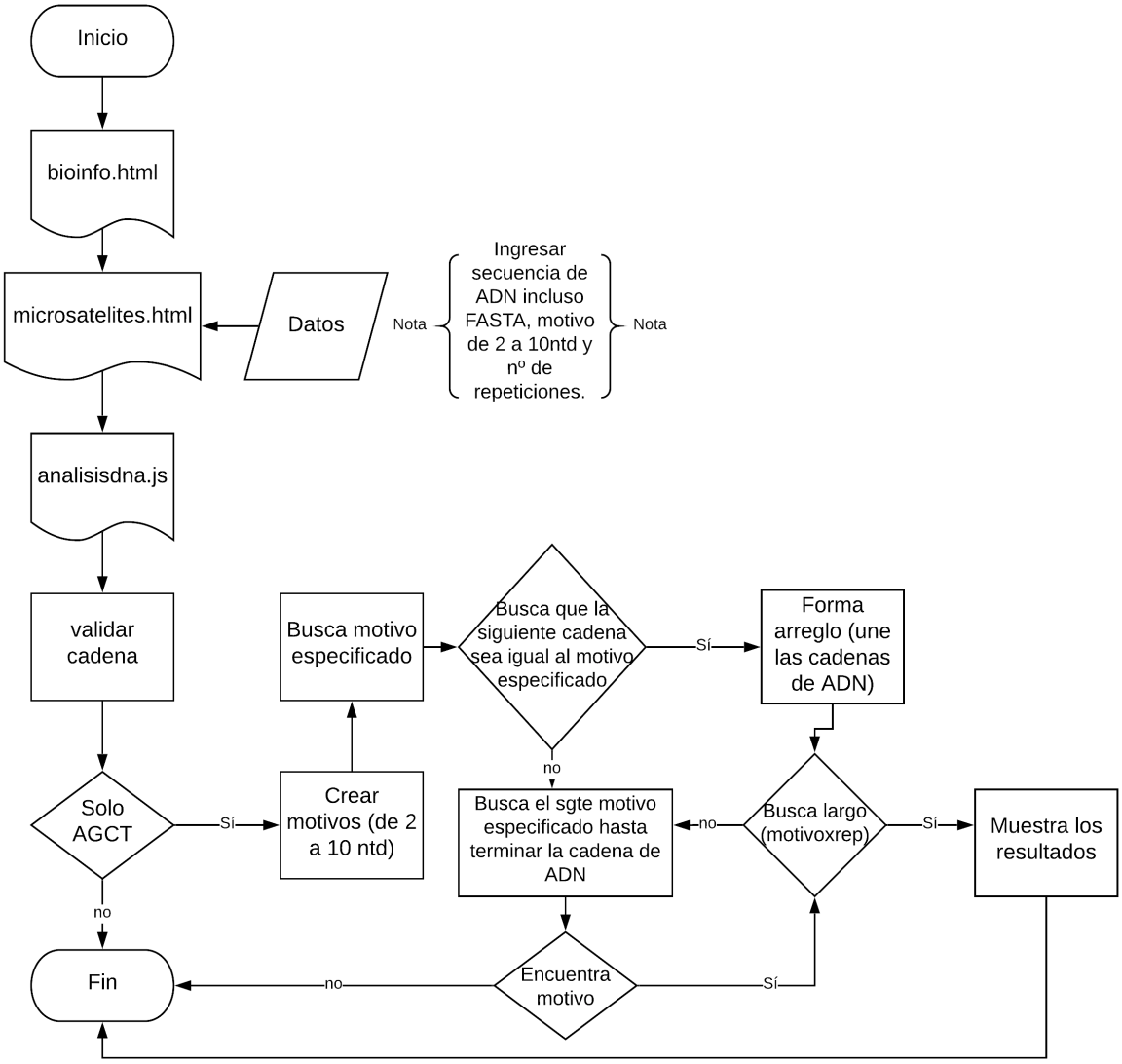


Figura 4.6.1. Diagrama de flujo del algoritmo para encontrar motivos microsatélites

Se colocaron varias secuencias para probar el buen funcionamiento de la herramienta bioinformática (archivo microsatélites.html), en este ejemplo colocamos la siguiente secuencia:

ATCGAGCTAGGTCGAAATAGTAGTAGTAGTAGTAGTCGATGAGCTG

La parte subrayada es el control, ya que es un trinucleótido (AGT) repetido 6 veces, empieza a partir de la base 18 hasta la 36 y fue detectado por el *script* como se ve en la figura 4.6.2. Cabe recalcar que se probó el *script* incluso con secuencias FASTA, obteniendo resultados satisfactorios.

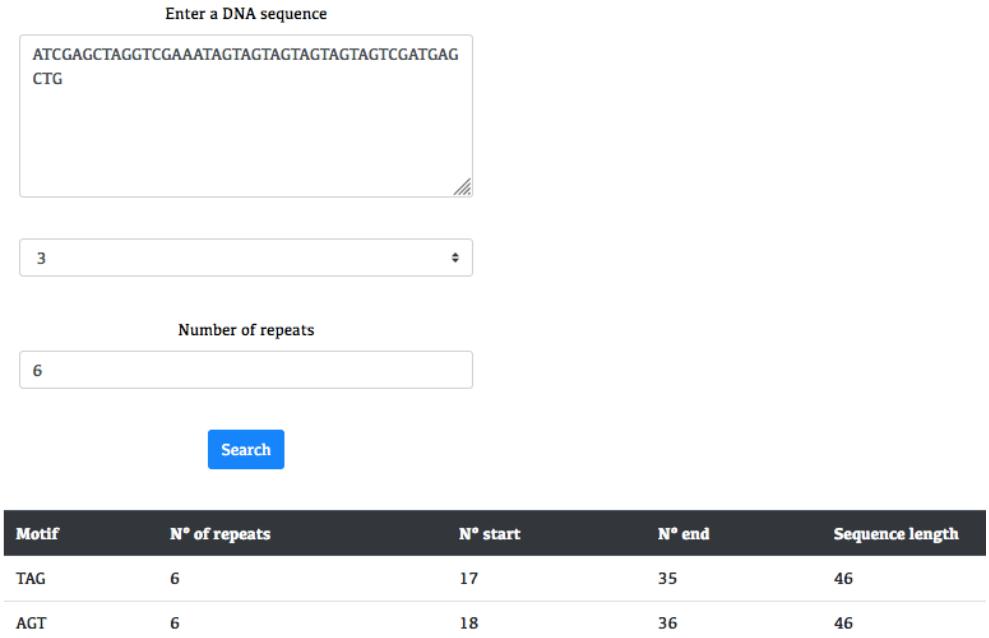


Figura 4.6.2. Se muestra el archivo "microsatelites.html" funcionando correctamente.

Como se puede apreciar, el *script* arroja el motivo control y otro motivo adicional. Además del nucleótido donde empieza la repetición y donde termina. Por tanto, si quisiera realizar un experimento de microsatélites bajo las condiciones del ejemplo, verificaría que el genotipo WT o región no tumoral tuviera 6 repeticiones de "AGT" y no más, por lo que, diseñaría un primer forward que va del nucleótido 1 al 18 y un reverso complementario que puede ir desde la región 36 en adelante. Si el amplicón en el genotipo mutante o región tumoral varía en tamaño en comparación al WT, esto indicaría una inestabilidad de microsatélites, como ocurre en cáncer colorrectal y glioblastoma. Es por ello que la herramienta es muy útil para detectar motivos repetidos de tipo microsatélite.

B) *Vector en fase*.- uno de los experimentos más comunes en biología molecular es la producción de proteínas recombinantes. Para ello, se debe insertar dentro de un plásmido (vector), el gen de interés, para poder hacerlo se usan tijeras moleculares (también llamadas enzimas de restricción) que cortan al plásmido circular en regiones específicas. Posteriormente, se deben cortar los extremos del gen de interés, luego colocar éste dentro del vector y

unir ambos fragmentos con una enzima llamada ligasa. Para ello, debemos tomar en cuenta que el gen debe estar en fase, es decir que al dividir la secuencia de tres en tres, estos tripletes codifiquen aminoácidos y una proteína funcional y no una trunca.

En el documento “fase.html” se coloca el código de lo que se verá en la web y se lo enlaza al documento escrito en JavaScript “analisistripletes.js” que contiene el código fuente que separa la cadena de ADN cada tres bases nitrogenadas. Los códigos HTML, CSS y JavaScript completos y comentados para desarrollar esta herramienta, se encuentran en el **anexo 6.4.**

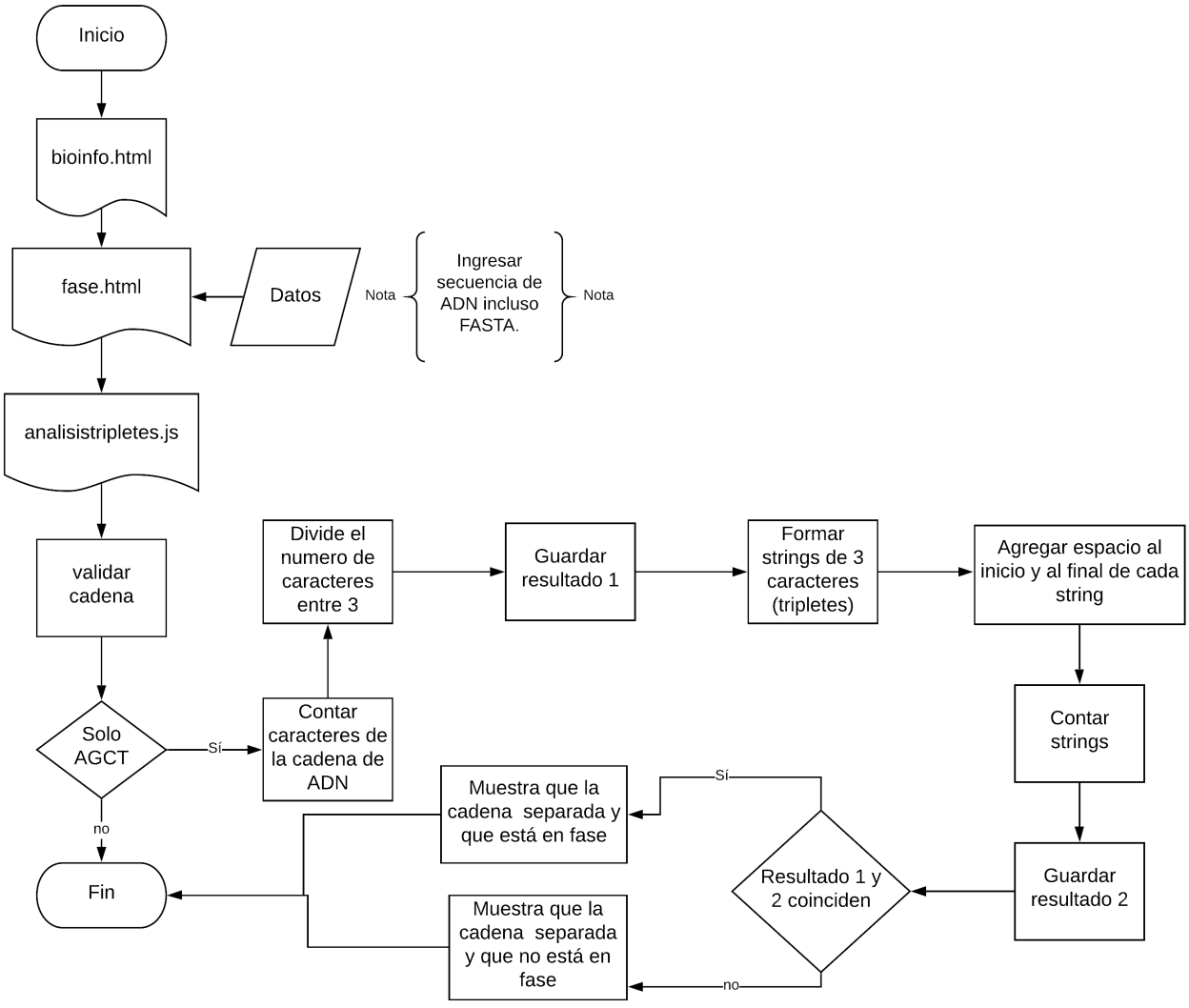


Figura 4.6.3. Diagrama de flujo del algoritmo para encontrar si la secuencia de ADN está en fase

En la figura 4.6.3 se muestra el funcionamiento de esta herramienta. El archivo “análisistripletes.js” toma la cadena de ADN ingresada por el usuario en el archivo “fase.html” y valida que los caracteres ingresados correspondan a las bases nitrogenadas, cuenta los caracteres, divide el número de caracteres entre tres, guardándolos como resultado 1. Posteriormente, divide a la cadena de ADN en *strings* de tres caracteres para formar los tripletes, agregando espacios al inicio y al final de cada triplete. Luego, cuenta los tripletes y genera el resultado 2. Si el resultado 1 coincide con el resultado 2, la cadena está en fase, caso contrario, no lo está.

A manera de ejemplo, para probar el buen funcionamiento del *script* se ingresó la siguiente secuencia compuesta por 46 bases nitrogenadas:
“ATCGAGCTAGGTCGAAATAGTAGTAGTAGTAGTAGTCGATGAGCTG”
Por lo que es de esperar que el resultado sea que la secuencia no está en fase y que tiene 46/3= 15.33 tripletes sobrando una base (figura 4.6.4).
Cabe recalcar el *script* se probó con muchas otras secuencias e incluso con secuencias FASTA para verificar su buen funcionamiento.

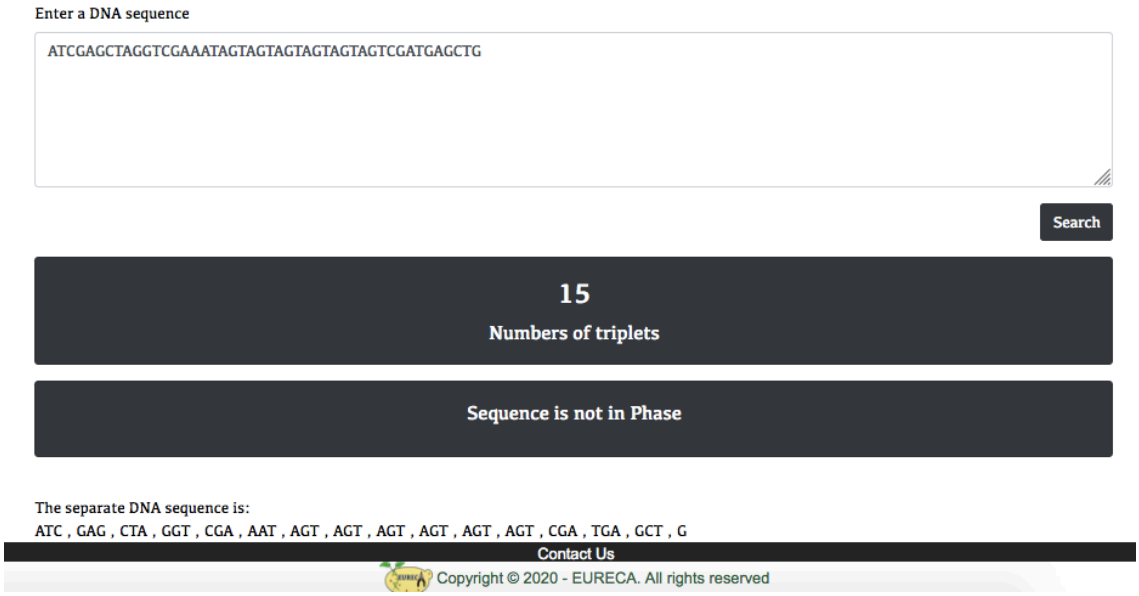


Figura 4.6.4. Se muestra el archivo “fase.html” funcionando correctamente.

Adicionalmente, se observa que en los recuadros negros figura el número de tripletes de la secuencia (15), se muestra la secuencia de ADN dividida de tres en tres sobrando un nucleótido (por tanto 16 separaciones) y como ambos números no coinciden se corrobora que la secuencia no está en fase (figura 4.6.4). Esta herramienta será de mucha utilidad para biólogos moleculares que pretendan realizar ingeniería genética.

C) *Análisis de sitios de restricción*.- cuando se tiene la secuencia del gen de interés que se quiere clonar, es necesario saber si existe alguna enzima de restricción capaz de cortar ese fragmento, puede hacerse de forma manual, buscando en los catálogos donde la enzima realiza la escisión, sin embargo, eso tomaría demasiado tiempo, por tal motivo se creó esta herramienta bionformática que permite detectar las enzimas que realizan el corte del fragmento dado.

Para ello se creó el archivo “restricción.html” que usa los documentos en JavaScript “enzimas.js” que es una base de datos que contiene el nombre de todas las enzimas de restricción con los cortes especificados y “analisisenzimas.js” que tiene la lógica de programación para reconocer qué enzima y donde realiza el corte. Los códigos HTML, CSS y JavaScript completos y comentados para desarrollar esta herramienta, se encuentran en el **anexo 6.5.**

En la figura 4.6.5 se muestra el funcionamiento de esta herramienta. El archivo “analisisenzima.js” valida el código de ambigüedad de las bases nitrogenadas (descrito en el anexo 6.5), busca la primera letra del motivo anterior a la escisión, si este motivo es igual a una base nitrogenada, se obtiene el motivo anterior al corte (resultado 1), caso contrario se busca que la siguiente letra coincida con la segunda letra del motivo declarado, esto se repite hasta llegar al largo del motivo anterior al corte. De esta forma se crea una subcadena, la cual es comparada con el largo del motivo anterior al corte declarado en la base de datos. Si coinciden se obtiene el resultado 1. Luego, se repite lo mismo pero con el motivo posterior a la escisión, con lo que se obtiene el resultado 2. Si el resultado 1 y 2 coinciden con lo declarado en la

base de datos de “enzimas.js”, se muestra el nombre de la enzima, el motivo anterior y posterior al corte y la posición en la que ocurrió la escisión.

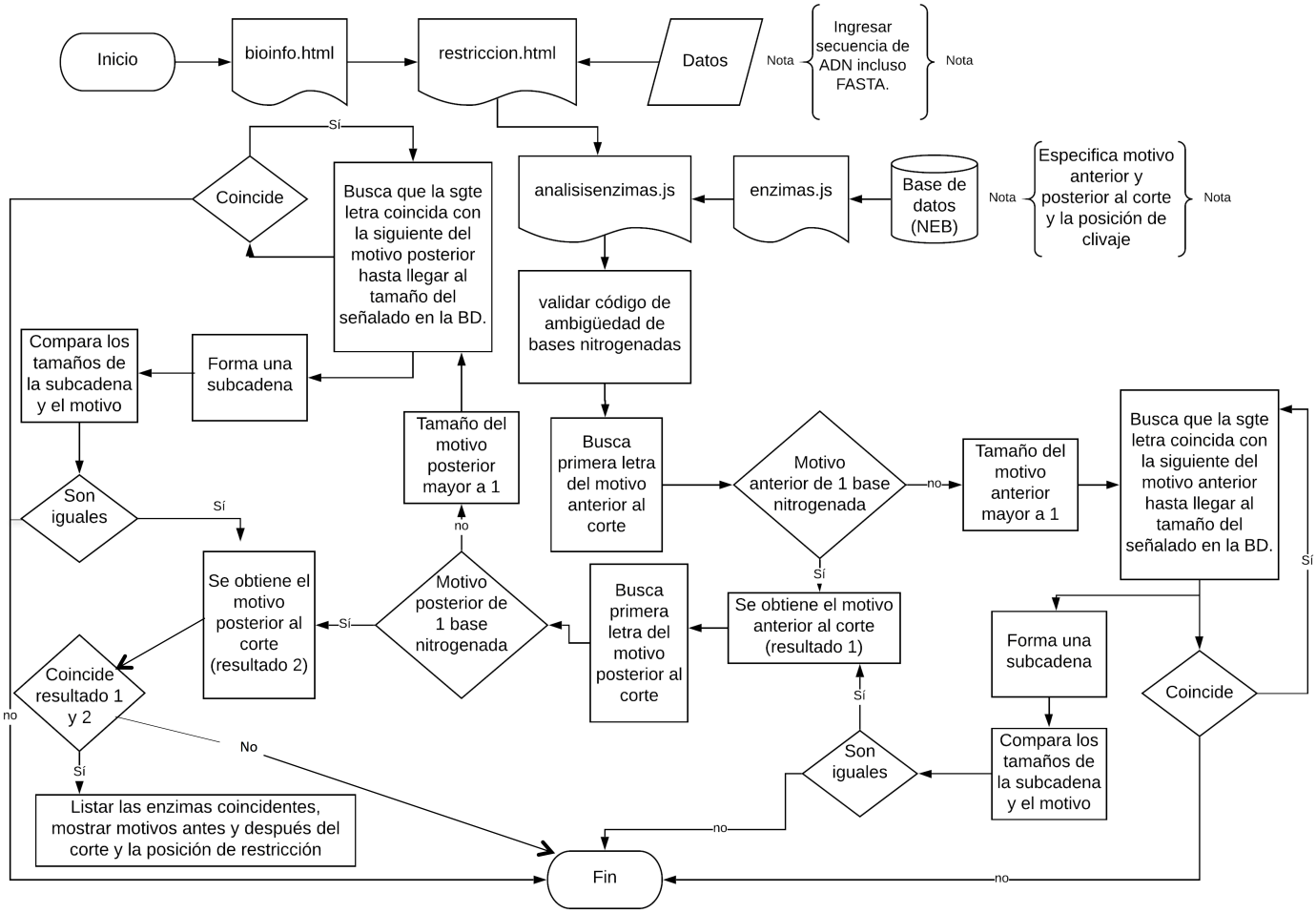


Figura 4.6.5. Diagrama de flujo del algoritmo para encontrar sitios de restricción

Para probar que el *script* funcione como esperamos, colocamos una secuencia con corte conocido como la siguiente: “AGGCCTCAGTCGGCTTTGAATTCGAGCGTGAGCGTGACGTGAGC” donde el motivo subrayado (G|AATTC) es reconocido y clivado por EcoRI, por lo que, una de las enzimas de la tabla de resultados debe ser EcoRI. Al colocar dicha secuencia en la página web, se observa que la enzima número 8 corresponde a EcoRI y se especifica la secuencia antes del corte y luego del corte, así como

la posición de escisión (figura 4.6.6). Cabe recalcar que el *script* también se probó con otras secuencias incluidas las FASTA.

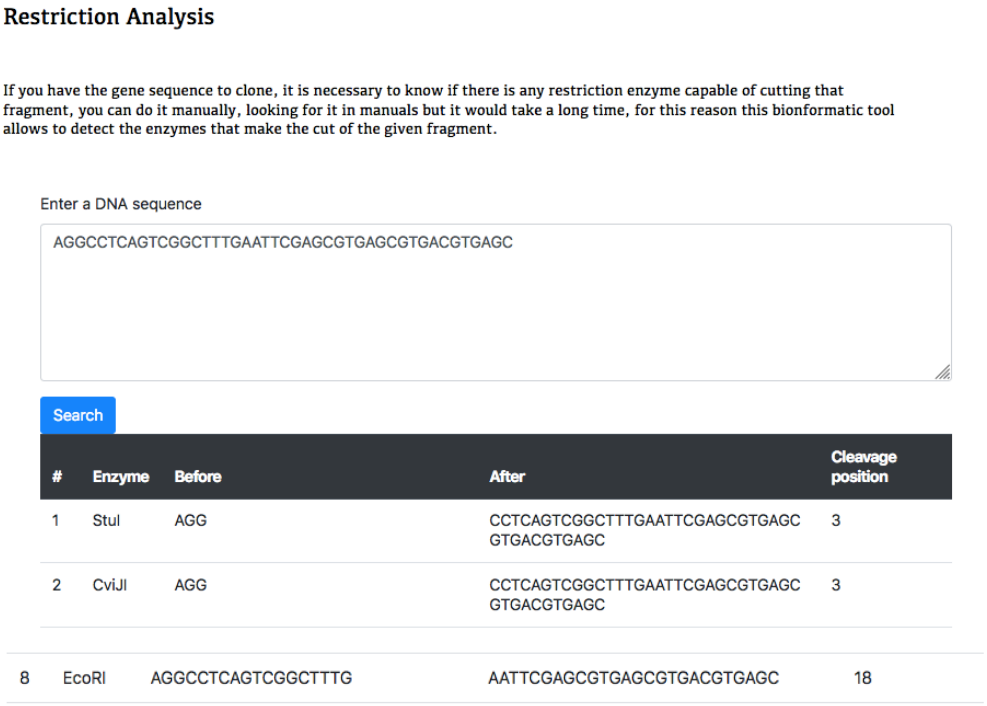


Figura 4.6.6. Se muestra el archivo "restriccion.html" funcionando correctamente.

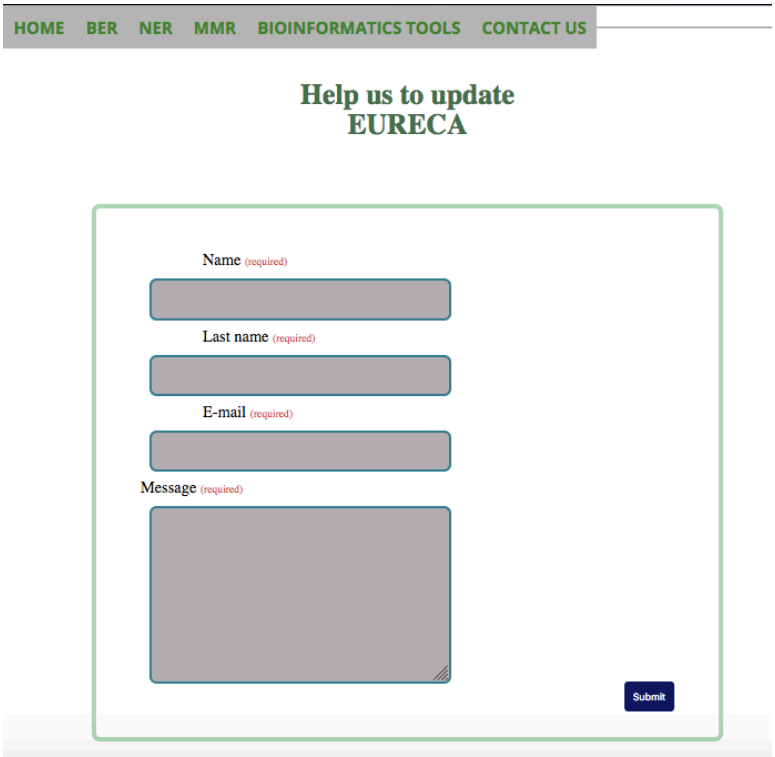
4.7. Construcción de la base de datos en PHPmyadmin

PHPmyadmin es una herramienta escrita en PHP que permite manejar la administración de MySQL a través de páginas web, usando internet. Se puede crear, eliminar bases de datos; crear, eliminar y alterar tablas; borrar, editar campos y obviamente ejecutar sentencias SQL. La combinación óptima es cuando se trabaja en un entorno de texto con Linux, apache, mysql-php, pero en este caso se usó un entorno gráfico como PHPmyadmin, por ser de fácil manejo. Una vez creada la base de datos dentro de "Eureca" se insertaron las quince tablas, con sus respectivos atributos y las futuras FKs, se identificaron los tipos de dato, así como las PKs. El código SQL para la construcción de la BD, las capturas de pantalla de las tablas creadas y el código SQL para formar las relaciones de las tablas se muestran en el **anexo 6.6** y el modelo relacional de la BD en PHPmyadmin en el **anexo 6.7.**

4.8. Construcción y validación del formulario de contacto

Cuando se habla de validaciones de formularios, lo primero que se piensa es en JavaScript, ya que uno de los motivos por los cuales se ha creado este lenguaje es para evitar el envío de información inútil al servidor, sin embargo, las validaciones *front-end* en el navegador pueden ser vulneradas fácilmente, si el usuario tiene JavaScript desactivado o bien, si se omitieron las validaciones. Por otro lado, las validaciones *back-end* con PHP son mucho más seguras. En este trabajo se validó *back-end*. Los códigos HTML, CSS y PHP para la construcción y validación del formulario, se encuentran en el **anexo 6.8.**

Al entrar en localhost/dashboard/eurecaweb/contacto.php se muestra el siguiente formulario de contacto.



Como se puede apreciar, se requiere que cada campo sea completado para que sea enviado al correo.

4.9. Conexión y consultas en la base de datos (prueba del sistema)

El código para conectar la BD y realizar las consultas se encuentra disponible en el **anexo 6.9.** Adicionalmente, se adjunta un video con el funcionamiento de toda la página web con datos ficticios que luego serán

reemplazados con los reales y está disponible en https://www.youtube.com/watch?v=v-qxPRO1ssQ

4.10. Almacenamiento en el hosting:

Hasta ahora, EURECA ha sido creada y validada por medio de consultas SQL en el servidor local, sin embargo, para que ésta se encuentre disponible a los investigadores es necesario subirla a un *hosting*. Para ello, se necesita de un dominio que puede ser gratuito o de pago. El problema de los dominios gratuitos es que tienen otras terminaciones como por ejemplo www.eurecabd.freetzi.com esto dificulta que el usuario recuerde la URL. En este caso, se compró el dominio www.eurecabd.com en namecheap. Como se puede apreciar en la figura 4.10.1 cuando el dominio es comprado, este se reserva hasta que se suban la totalidad de archivos que conforman la página web.

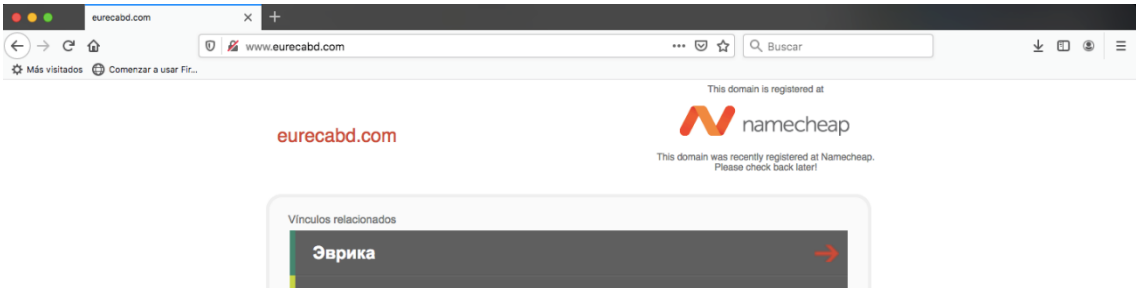


Figura 4.10.1. Pantalla que muestra que el dominio de EURECA ya fue registrado

Los *hostings* también pueden ser gratuitos o de pago. Para subir los archivos HTML, CSS, JavaScript, PHP y SQL programados para EURECA, se requiere de un protocolo FTP por sus siglas en inglés file-transfer Protocol (protocolo de transferencia de ficheros), el cual permite subir y descargar archivos entre la computadora del usuario (cliente FTP) y la computadora que sirve las páginas web (servidor FTP). Este es uno de los diversos protocolos de la red Internet y es ideal para transferir grandes bloques de datos por la red. Las páginas web son en su mayoría subidas a los servidores mediante este protocolo. Aquí se usó Filezilla por ser un cliente FTP gratuito. Se contrató un *hosting* gratuito tal como se describe en

materiales y métodos y que nos sirve para subir nuestros archivos a la web. En este trabajo no se liberaron los archivos, hasta que el artículo científico producto de esta investigación sea aceptado.

4. CONCLUSIONES

- Se creó la base de datos relacional dentro de la plataforma web EURECA (EUkaryote DNA REpair CApacity), a partir de datos curados provenientes de al menos cien artículos científicos revisados y consultas bioinformáticas sobre datos de genes y proteínas de los sistemas de reparación indirectos del ADN (BER, NER y MMR) que pertenecen a organismos eucariotas (plantas, levaduras y humanos).
- Se realizó una página web para EURECA completamente funcional, por el momento disponible en el servidor local.
- Se desarrollaron tres herramientas bioinformáticas para la página web de EURECA que permiten encontrar motivos microsatélites; determinar si un vector se encuentra en fase, separando de tres en tres las bases nitrogenadas y realiza el análisis de enzimas de restricción de forma automática.
- EURECA es una herramienta muy importante para biólogos moleculares que realicen investigaciones en sistemas de reparación indirecta del ADN tanto en levaduras, plantas como en humanos y ayudará a disminuir la confusión entre los diferentes nombres de las proteínas, no redundar en las investigaciones y así lograr una gestión más eficiente de los datos existentes.

5. BIBLIOGRAFÍA

1. Apache. Revisado en: https://www.apachefriends.org/es/index.html
2. Bilichak A, Ilnystkyy Y, Hollunder J, Kovalchuk I. 2012. The progeny of

Arabidopsis thaliana plants exposed to salt exhibit changes in DNA methylation, histone modifications and gene expression. PLoS One 7:e30515.

3. Camps Paré P, Casillas Santillán LA. 2005. Base de datos. Editorial Fundació per a la Universitat Oberta de Catalunya. ISBN 84-9788-269-5, Barcelona, España.
4. Campus MVP. Revisado en: https://www.campusmvp.es/recursos/post/Que-son-las-Aplicaciones-Web-Progresivas-o-Progressive-Web-Apps.aspx
5. Chirinos-Arias M y Jiménez J. 2015. Transference of some microsatellite molecular markers from Fabaceae family to Andean Lupin (*Lupinus mutabilis* Sweet). Sci Agropecua 6(1): 51-58.
6. Chirinos-Arias MC, Spampinato, C.P. 2020. Growth and development of AtMSH7 mutants in Arabidopsis thaliana. Plant Physiol. and Bioch. 146: 329-336.
7. Chirinos-Arias MC, Spampinato, C.P. 2021. Role of the mismatch repair protein MSH7 in Arabidopsis adaptation to acute salt stress. Plant Physiol. and Bioch. 169: 280-290.
8. El sistema operativo GNU y el movimiento del software libre. Revisado en: https://www.gnu.org/licenses/licenses.es.html
9. Erie DA, Weninger KR. 2014. Single molecule studies of DNA mismatch repair. DNA Repair, 20: 71-81.
10. Förster K. 2008. HTML5 guidelines for web developers. Pearson Education, Boston, USA.
11. Friedhoff P, Li P, Gotthardt J. 2016. Protein-protein interactions in DNA mismatch repair. DNA Repair, 38: 50-57.
12. Gasteiger E, Hoogland C, Gattiker A, Duvaud S, Wilkins MR, Appel RD, Bairoch A. Protein Identification and Analysis Tools on the ExPASy Server; (In) John M. Walker (ed): The Proteomics Protocols Handbook, Humana Press (2005). pp. 571-607.
13. Gauchat D. 2012. El gran libro de HTML5, CSS3 y JavaScript. Ediciones técnicas marcombo, España.

14. Gliffy: Diagramming Software and Team Collaboration Tools. Revisado en: https://www.gliffy.com/
15. Gómez R, Spampinato CP. 2013. Mismatch recognition function of Arabidopsis thaliana MutSγ. DNA Repair 12: 257-264.
16. Groth A, Ray-Gallet D, Quivy JP, Lukas J, Bartek J, Almouzni G. 2005. Human Asf1 regulates the flow of S phase histones during replicational stress. Mol Cell 17(2): 301-311.
17. Hirouchi T, Nakajima S, Najrana T, Tanaka M, Matsunaga T, Hidema J, y col. 2003. A gene for a class II DNA photolyase from *Oryza sativa*: cloning of the cDNA by dilution-amplification. *Mol. Genet. Genomics,* 269: 508–516.
18. Hoeijmakers JH. Genome maintenance mechanisms for preventing cancer. Nature 2001; 411(6835):366-74.
19. Iyama T, Wilson DM. 2013. DNA repair mechanisms in dividing and non-divinding cells. DNA Repair, 12(8):620-636.
20. Jiricny J. 2013. Postreplicative mismatch repair. Cold Spring Harb Perspect Biol, 5: a012633.
21. Kadyrov FA, Dzantiev L, Constantin N, Modrich P. 2006. Endonucleolytic function of MutLalpha in human mismatch repair. Cell 126(2): 297-308.
22. Krokan HE, Bjoras M. 2013. Base Excision Repair. Cold Spring Harb Perspect Biol, 5(4):a012583.
23. Kunkel TA, Erie DA. 2015. Eukaryotic mismatch repair in relation to DNA replication. Annu Rev Genet, 49: 291-313.
24. Kunkel TA. 2015. Celebrating DNA's Repair Crew. Cell, 163: 1301-1303.
25. Lario LD, Botta P, Casati P, Spampinato CP. 2015. Role of AtMSH7 in UV-B- induced DNA damage recognition and recombination. J Exp Bot 66: 3019- 3026.
26. Lathrop L. 1999. An Indexer's Guide to the Internet. 2nd Edition American Society Indexer, Phoenix, AZ, USA.
27. Leibeling D, Laspe P, Emmert S. Nucleotide excision repair and cancer. J Mol Histol 2006; 37(5- 7):225-38.
28. Leonard JM, Bollmann SR, Hays JB. 2003. Reduction of stability of

Arabidopsis genomic and transgenic DNA repeat sequences (Microsatellites) by inactivation of AtMSH2 mismatch-repair function. Plant Physiol 133: 328-338.

29. Li F, Ortega J, Gu L, Li GM. 2016. Regulation of mismatch repair by histone code and posttranslational modifications in eukaryotic cells. DNA Repair, 38: 68-74.
30. Li GM. Modrich P. 1995. Restoration of mismatch repair to nuclear extracts of H6 colorectal tumor cells by a heterodimer of human MutL homologs. Proc Natl Acad Sci USA 92(6): 1950-1954.
31. Liu F, Xu Y, Zhou L, Ali A, Jiang H, Zhu S, Li X. 2019. DNA Repair Gene *ZmRAD51A* Improves Rice and *Arabidopsis* Resistance to Disease. Int J Mol Sci, 13: 20(4):807.
32. Lloyd AH, Milligan AS, Langridge P; Able JA. 2007. *TaMSH7*: A cereal mismatch repair gene that affects fertility in transgenic barley (*Hordeum vulgare* L.) *BMC Plant Biology* 7:67
33. Luján-Mora S. 2002. Programación de aplicaciones web: Historia, principios básicos y clientes web. Editorial Club Universitario ISBN 8484542968. España.
34. Manova V, Georgieva M, Borisov B, Stoilova B, Gecheff K, Stoilov L. 2009. Genomic and gene-specific induction and repair of DNA damage in barley, in *Induced Plant Mutations in the Genomics Era*, ed. Q. Y. Shu (Rome: Food and Agriculture Organization of the United Nations), 133–136.
35. MAMP & MAMP PRO. Revisado en: https://www.mamp.info/en/
36. Marqués M. 2011. Bases de datos. Publicacions de la Universitat Jaume I. Castelló de la Plana, España.
37. Mellon I, Rajpal DK, Koi M, Boland CR, Champe GN. 1996. Transcription- coupled repair deficiency and mutations in human mismatch repair genes. Science 272(5261): 557-560.
38. Millán ME. 2017. ISBN PDF 978-958-765-487-5. Fundamentos de bases de datos. Programa Editorial Universidad del Valle. Colombia.

https://www.ibm.com/developerworks/ssa/data/library/tipos_bases_de_datos/index.html
39. MySQL. Revisado en: https://www.mysql.com/
40. National Center for Biotechnology Information (NCBI). Revisado en: https://www.ncbi.nlm.nih.gov/
41. NEB Tools. Revisado en: https://enzymefinder.neb.com/#!/#nebheader
42. Niederst Robbins J. 2012. Learning Web Design A beginner's guide to HTML, CSS, JavaScript and web graphics. 4ta edición O´Reilly, Canadá.
43. O'Reilly, Printing Strings. Revisado en: https://www.oreilly.com/library/view/programming-php-3rd/9781449361068/ch04s02.html
44. PHP. Revisado en: http://php.net/manual/es/language.types.intro.php
45. Quaite FE, Takayanagi S, Ruffini J, Sutherland JC, Sutherland BM. 1994. DNA damage levels determine cyclobutyl pyrimidine dimer repair mechanisms in alfalfa seedlings. *Plant Cell*, 6: 1635–1641.
46. Saccharomyces Genome Database (SGD). Revisado en: https://www.yeastgenome.org/
47. Spampinato CP, Gomez RL, Galles C, Lario LD. 2009. From bacteria to plants: a compendium of mismatch repair assays. Mutat Res, 682: 110-128.
48. Spivak G. 2015. Nucleotide excision repair in humans. DNA repair, 36: 13-18.
49. Stapleton AE, Thornber CS, Walbot V. 1997. UV-B component of sunlight causes measurable damage in field-grown maize (*Zea mays* L.): developmental and cellular heterogeneity of damage and repair. *Plant Cell Environ,* 20: 279–290.
50. Sun XL, Yang T, Guan, JP, Ma Y, Jiang JY, Cao R, Burlyaeva M, Vishnyakova M, Semenova E, Bulyntsev S, Zong XX. 2012. Development of 161 novel EST-SSR markers from Lathyrus sativus (Fabaceae). American Journal of Botany 99: 379-390.
51. Tafurt Y, Marin MA.2014. Principales mecanismos de reparación de daños en la molécula de ADN. Revista Biosalud, 13(2): 95-110.

52. Takeuchi Y, Murakami M, Nakajima N, Kondo N, Nikaido O. 1998. The photorepair and photoisomerization of DNA lesions in etiolated cucumber cotyledons after irradiation by UV-B depends on wavelength. *Plant Cell Physiol,* 39: 745–750.
53. Tam S, Samipak S, Britt A, Chetelat R. 2009. Characterization and comparative sequence analysis of the DNA mismatch repair MSH2 and MSH7 genes from tomato. Genetica, 137: 341-354.
54. The Arabidopsis Information Resouce (TAIR). Revisado en: https://www.arabidopsis.org/
55. Taylor RM, Nikaido O, Jordan BR, Rosamond J, Bray CM, Tobin AK. 1996. Ultraviolet-B-induced DNA lesions and their removal in wheat (*Triticum aestivum* L.) leaves. *Plant Cell Environ,* 19: 171–181.
56. Uniprot Knowlegebase (UniProtKB). Revisado en: https://www.uniprot.org/
57. W3Schools. Revisado en: https://www.w3schools.com/php/default.asp
58. WampServer, la plate-forme de développement Web sous Windows. Revisado en: http://www.wampserver.com/en/
59. Yamamoto A, Tanbir N, Hirouchi T, Teranishi M, Hidema J, Morioka H, y col. 2008. Temperature-sensitive photoreactivation of cyclobutane thymine dimer in soybean. *J. Radiat. Res.,* 49: 189–196.
60. Yoshihara R, Imaki T, Hori M, Watanabe C, Yamamoto K, Takimoto K. 2005. CPD photolyase gene from *Spinacia oleracea*: repair of UV-damaged DNA and expression in plant organs. *J. Radiat. Res.,* 46: 157–164.
61. Zhang Z, Shibahara K, Stillman B. 2000. PCNA connects DNA replication to epigenetic inheritance in yeast. Nature 408(6809): 221-225.

6. ANEXOS

ANEXO 6.1: Código HTML de la página web.

El archivo principal, del cual las demás pestañas parten es **index.html** que en este trabajo es igual a **inicio.html**, cuyo código se muestra a continuación:

```
1  <html lang="es">
2          <link rel="stylesheet" href="css/encabezado.css">
3          <meta name="tipo" content="text/html;" http-equiv="content-type" charset="utf-8">
4      <div id="arriba" align="center">
5          <img src="imagenes/logonom.png" height="140" width="290" title="Eureca">
6          <h2 align="center">Eukaryote DNA Repair Capacity</h2>
7      </div><br><br><br>
8      <meta name="tipo" content="text/html;" http-equiv="content-type" charset="utf-8">
9      <center>
10         <img src="imagenes/portada.png" height="350" width="1240">
11     </center>
12     <head>
13     <body>
14         <div id="header">
```

Por otro lado, en el navegador (<nav>) se colocaron las pestañas que direccionan a las ventanas BER (línea 18), NER (línea 19), MMR (línea 20), herramientas bioinformáticas (línea 21) y el formulario de contacto (línea 22), siempre mediante el atributo href.

```
14         <div id="header">
15             <h3>
16                 <ul class="nav">
17                     <li><a href="inicio.html">HOME</a></li>
18                     <li><a href="ber.html" title="ber">BER</a></li>
19                     <li><a href="ner.html" title="ner">NER</a></li>
20                      <li><a href="mmr.html" title="mmr">MMR</a></li>
21                      <li><a href="bioinfo.html" title="bioinfo">BIOINFORMATICS TOOLS</a></li>
22                      <li><a href="contacto.php" title=" contacto">CONTACT US</a></li>
23                     </li>
24                     <br>
25                     <hr align="CENTER" size="2" width="250" color="#b7b7b7" noshade></hr>
26                 </ul>
27             </h3>
28         </div>
```

Se colocó el código CSS dentro del archivo “index.html” para darle color al cuerpo (body). Si bien se pudo enlazar a un archivo .css o agregar en encabezado.css, era necesario dividir en apartados y resultó más fácil colocarlo en el mismo archivo html.

```
30      <body bgcolor="white">
31      <div id="dos">
32      <table width="75%">
33          <tr>
34              <td>
35              </td>
36              <td><font face="tahoma" color="black">
37                  <img src="imagenes/logonom.png" height="100" width="200">
38                  <br>
39                  <p><h2><font face="Arial" color="#008c2e"><b>About EURECA</b></h2></font></p>
40                  <br>
41                  <p><h4>Living organisms are continuously exposed to endogenous stress (for exam
42                  </h4><br>
43                  <img src="imagenes/inicio.png" height="300" width="450">
44              </td>
45          </tr>
46      </table>
47  </div>
```

```
48  <style type="text/css">
49      #dos{
50      margin:auto;
51      width:1240;
52      font-family:Arial, Helvetica, sans-serif;
53      background-color:white;
54      }
55  </style>
56  </body>
57      </body>
```

En el pie de página (*footer*) se colocó un enlace para contacto mediante el atributo “href”. Se insertó el logo en miniatura y los *copyrights*. El estilo del *footer* también se colocó en el mismo HTML con código CSS como se puede ver en la figura de abajo (línea 70 a la 77).

```
60          <div id="footer" style="color:white;" align="center">
61              <ul>
62                  <a href="mailto:chirinos@cefobi-conicet.gov.ar" target="_blank"><font face="Arial" color="
63                  </ul>
64          </div>
65
66          <div align="center">
67              <img src="imagenes/logonom.png" width="55px" height="35px">
68              <font face="Arial" color="#00703c"> Copyright © 2020 - EURECA. All rights reserved</font></a>
69          </div>
70          <style>
71              #footer {
72              margin:auto;
73              width:1240px;
74              height:20px;
75              background: #242424;
76              }
77          </style>
78      </body>
79  </html>
```

El archivo “index.html” fue usado como base para las pestañas “BER”, “NER” y “MMR” solo que dentro del HTML se incluyó un código que hace que al darle click a las figuras de planta, humano o levadura, se pueda acceder a estas pestañas. A manera de ejemplificar, solo se coloca el archivo “**ner.html**”, ya que es el mismo código adicional que se incluyó en “ber.html” y “mmr.html”.

```
    <div align="left">
        <h3><font face="Arial" color="#008c2e">SELECT ONE</font></h3>
    <a href="human.html"><img src="imagenes/human.png" align="left" height="180" width="130"></a>
    <a href="yeast.html"><img src="imagenes/yeast.png" align="left" height="140" width="150"></a>
    <a href="arabidopsis.html"><img src="imagenes/arabidopsis.png" align="left" height="160" width="150"></a>
</div>
    <br>
```

Lo mismo ocurre con "**bioinfo.html**" que es la pestaña de las herramientas bioinformáticas, solo que aquí se enlaza (por medio de href) a imágenes que tienen enlace a las pestañas de búsqueda de microsatélites (microsatelites.html), secuencia en fase (fase.html) y análisis de sitios de restricción (restricción.html).

```
    <div align="left">
        <h3><font face="Arial" color="#008c2e">SELECT ONE</font></h3>
    <a href="microsatelites.html"><img src="imagenes/microsatelites.png" align="left" height="150" width="150"></a>
    <a href="fase.html"><img src="imagenes/fase.png" align="left" height="150" width="150"></a>
    <a href="restriccion.html"><img src="imagenes/restriccion.png" align="left" height="150" width="150"></a>
</div>
    <br>
```

ANEXO 6.2: Código CSS de la página web

6.2.1. "encabezado.css"

A continuación, se muestra el código CSS usado para el archivo "encabezado.css". En este código * { margin: 0; padding: 0; } elimina el margen y el relleno de todos los elementos HTML.

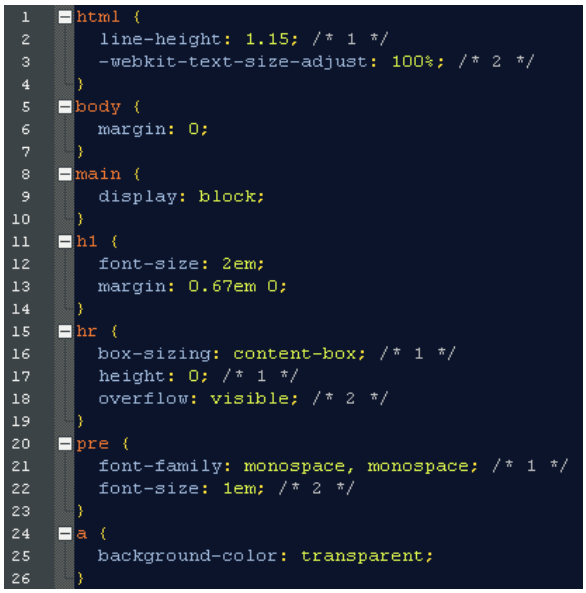
```
*{
    margin:0px;
    padding: 0px;
}
```

```
#header{
    margin: auto;
    width: 1240px;
    font-family: Arial, Helvetica, sans-serif;
}
ul, ol{
    list-style:none;
}
.nav > li {
    float: left;
}
.nav li a{
    background: #b7b7b7;
    color: #008c2e;
    align: height;
    text-decoration: none;
    font-family: Open Sans, Arial, sans-serif;
    padding: 10px 12px;
    display: block;
}
```

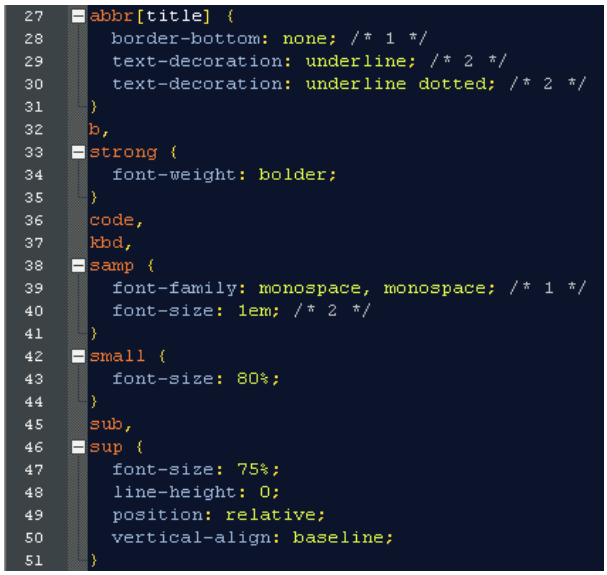
```
.nav li ul{
    display: none;
    position: absolute;
    min-width: 140px;
    z-index: 100;
}
.nav li:hover > ul{
    display:block;
}
.nav li ul li{
    position:relative;
}
.nav li ul li ul{
    right:-140px;
    top:0px;
}
```

6.2.2. “normalize.css”

Sin embargo, es necesario normalizar estilos con CSS3, lo que significa hacer que todos los navegadores web como Firefox, Chrome, Safari y Opera, puedan interpretar los estilos por defecto, ya que los estilos se implementan al último y pueden tener diferencias entre los mismos navegadores. Al contrario de algunos “CSS *reset”*, “CSS *normalize”* preserva los valores por defecto de los navegadores, corrige errores e inconsistencias de los mismos y normaliza estilos para una amplia gama de elementos, para ello en “estilos.css” se coloca @import url(‘normalize.css’) o en “index.html” se coloca <link rel="stylesheet" href="css/normalize.css">. Este código permite remover el margen en todos los navegadores, corrige el tamaño y tipo de la letra en las diferentes secciones o cajas del documento HTML que lo contiene. *Normalize* tiene el siguiente código:



```
html {
  line-height: 1.15; /* 1 */
  -webkit-text-size-adjust: 100%; /* 2 */
}
body {
  margin: 0;
}
main {
  display: block;
}
h1 {
  font-size: 2em;
  margin: 0.67em 0;
}
hr {
  box-sizing: content-box; /* 1 */
  height: 0; /* 1 */
  overflow: visible; /* 2 */
}
pre {
  font-family: monospace, monospace; /* 1 */
  font-size: 1em; /* 2 */
}
a {
  background-color: transparent;
}
```



```
abbr[title] {
  border-bottom: none; /* 1 */
  text-decoration: underline; /* 2 */
  text-decoration: underline dotted; /* 2 */
}
b,
strong {
  font-weight: bolder;
}
code,
kbd,
samp {
  font-family: monospace, monospace; /* 1 */
  font-size: 1em; /* 2 */
}
small {
  font-size: 80%;
}
sub,
sup {
  font-size: 75%;
  line-height: 0;
  position: relative;
  vertical-align: baseline;
}
```

```
sub {
  bottom: -0.25em;
}

sup {
  top: -0.5em;
}
img {
  border-style: none;
}
button,
input,
optgroup,
select,
textarea {
  font-family: inherit; /* 1 */
  font-size: 100%; /* 1 */
  line-height: 1.15; /* 1 */
  margin: 0; /* 2 */
}
button,
input { /* 1 */
  overflow: visible;
}
```

```
button,
select { /* 1 */
  text-transform: none;
}
button,
[type="button"],
[type="reset"],
[type="submit"] {
  -webkit-appearance: button;
}
button::-moz-focus-inner,
[type="button"]::-moz-focus-inner,
[type="reset"]::-moz-focus-inner,
[type="submit"]::-moz-focus-inner {
  border-style: none;
  padding: 0;
}
button:-moz-focusring,
[type="button"]:-moz-focusring,
[type="reset"]:-moz-focusring,
[type="submit"]:-moz-focusring {
  outline: 1px dotted ButtonText;
}
fieldset {
  padding: 0.35em 0.75em 0.625em;
}
```

```
legend {
  box-sizing: border-box; /* 1 */
  color: inherit; /* 2 */
  display: table; /* 1 */
  max-width: 100%; /* 1 */
  padding: 0; /* 3 */
  white-space: normal; /* 1 */
}
progress {
  vertical-align: baseline;
}
textarea {
  overflow: auto;
}
[type="checkbox"],
[type="radio"] {
  box-sizing: border-box; /* 1 */
  padding: 0; /* 2 */
}
[type="number"]::-webkit-inner-spin-button,
[type="number"]::-webkit-outer-spin-button {
  height: auto;
}
```

```
[type="search"] {
  -webkit-appearance: textfield; /* 1 */
  outline-offset: -2px; /* 2 */
}
[type="search"]::-webkit-search-decoration {
  -webkit-appearance: none;
}
::-webkit-file-upload-button {
  -webkit-appearance: button; /* 1 */
  font: inherit; /* 2 */
}
details {
  display: block;
}
summary {
  display: list-item;
}
template {
  display: none;
}
[hidden] {
  display: none;
}
```

Por otro lado, para hacer la página web "*responsive*", es decir, que se adapte también a la pantalla del celular, se incluyó este código en el archivo "encabezado.css"

```
@media (max-width:480px){
    header{
        width:100%;
        }
    }
```

ANEXO 6.3: Código (HTML, CSS y JavaScript) usado para el desarrollo de la herramienta que busca motivos microsatélites.

6.3.1. “microsatelites.html”

Se hizo una breve descripción de los microsatélites y de lo que trata la herramienta (línea 46 y 47 respectivamente).

```
44  <div align="left"><font face="kefa" color="black">
45      <b><p><h4>Microsatellites Motifs Search</p></h4><br></b>
46      <p><h6>The microsatellites are DNA sequences in which a frag
47      <p><h6>The following bioinformatics tool helps to find micro
48      <br>
49      <br>
50  </div>
```

Por otro lado, se enlazó este archivo anterior a “analisisdna.js”, el cual contiene el código fuente que encuentra los motivos microsatélites (línea 65). <body onload=”llenar motivos()”> (línea 70) es un elemento que ejecuta un script una vez que una página web se ha cargado completamente. Luego, se colocó un formulario (a partir de la línea 72), que es donde las personas pondrán la secuencia de ADN en un área de texto (línea 77, 78).

```
65      <script type="text/javascript" src="assets/js/analisisdna.js"></script>
66  </head>
67  <tr><tr><tr>
68      <div class="container">
69          <div class="row">
70              <body onload="llenarmotivos()">
71                  <div class="col-md-5">
72                      <form class="" id="formulario">
73                          <div class="form-group" align="center">
74                              <body>
75                                  <td><font face="kefa" color="black">
76                              <label for="secuencia_adn">Enter a DNA sequence</label>
77              <textarea style="height: 10em;" name="secuencia_adn" id="secuencia" class="form-control"
78                                      cols="30" rows="10" required></textarea>
79                                  </td>
80                                  </body>
```

El estilo de la tabla se da en la línea 84, se usa el elemento *select* que muestra un menú de opciones con los motivos que se elegirá, por ejemplo “2” representa un dinucleótido, “3” un trinucleótido, etc (línea 86). El atributo *disabled* (línea 87) es un booleano que especifica que el usuario no puede elegir otro valor que no esté especificado en el menú (id=motivo1), en este caso motivos que van de dos a diez nucleótidos. En la línea 92, colocamos dentro de type=“text” el número de repeticiones y como es un área de texto, se debe escribir un número y por último se incluyó el botón “search” (línea 97) para ejecutar el *script*.

```
83    <div class="form-group" align="center">
84    <link rel="stylesheet" type="text/css" href="node_modules/bootstrap/dist/css/bootstrap.min.css">
85        <td>
86            <select name="" class="custom-select" id="motivos">
87                <option selected disabled value="" id="motivo1">Select a motif-length</option>
88            </select>
89        </div>
90        <br>
91        <div class="form-group" align="center"><label for="repeticiones">Number of repeats</label>
92            <input class="form-control" type="text" name="repeticiones" id="repeticiones">
93        </div>
94        <br>
95        <div class="form-group" align="center">
96        <button type="button" class="btn btn-primary" id="encontrar"
97        onclick=" buscarcoincidencias()">Search</button>
98        </div>
99    </form>
```

```
102       <div class="col-md-5">
103           <div class="row-auto">
104               <!-- <p>Resultado:</p> -->
105               <div class="form-group" >
106                   <p id="salida"></p>
107               </div>
108           </div>
109           <div class="row-auto">
110               <p id="tripletes"></p>
111           </div>
```

Por último, se crea la tabla con los resultados que mostrarán el motivo, número de repeticiones, número de nucleótido donde empieza el microsatélite, donde termina y el largo de la secuencia (de la línea 119 a la 123).

```
114   <div class="row ">
115       <div class="table-responsive" align="center">
116           <table class="table" id="tabla_analisis">
117               <thead class="thead-dark">
118                   <tr>
119                       <th scope="col">Motif</th>
120                       <th scope="col">N° of repeats</th>
121                       <th scope="col">N° start</th>
122                       <th scope="col">N° end</th>
123                       <th scope="col">Sequence length</th>
124                   </tr>
125                   <br>
126               </thead>
127               <tbody id="tablainf">
128               </tbody>
129           </table>
```

Adicionalmente, se agregó la librería JQuery de JavaScript, la cual simplifica la tarea de programar y permite agregar interactividad en el sitio web mediante *plugins*.

```
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
    integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous">
    </script>
```

6.3.2. “analisisdna.js”

Como se mencionó anteriormente, la programación que permite realizar la búsqueda de microsatélites está en el archivo “analisisdna.js”, para ello, se usa la función “validarcadena” que toma un argumento llamado valor, se define la variable cadena y ese valor se divide por medio del método split(), con ello

se define la cadena de ADN ingresada (de la línea 1 a la 3). También se valida que los motivos no sean números (NaN) (de la línea 6 a la 10).

```
1  function validarcadena(valor) {
2      var cadena = valor.split("");
3      var centinela = true;
4
5
6      cadena.forEach(element => {
7          element = element.toLowerCase();
8          if (!isNaN(Number.parseInt(element, 10))) {
9              // console.log("false");
10             centinela = false;
11         }
12     });
```

Por otro lado, se especifica que las letras que se deben reconocer son mayúsculas y son las correspondientes a las bases nitrogenadas adenina (A), guanina (G), citosina (C) y timina (T). Con esto se establece la naturaleza de la cadena de ADN conformada por letras y con “return centinela;” se indica que si las validaciones están bien, se retorne “verdadero”, caso contrario retorne falso, con lo cual no se ejecutará el *script* (de la línea 14 a la 29).

```
14     cadena.forEach(element => {
15         element = element.toUpperCase();
16         if (element != "A") {
17             if (element != "G") {
18                 if (element != "C") {
19                     if (element != "T") {
20                         //console.log("false");
21                         centinela = false;
22                     }
23                 }
24             }
25         }
26     });
27     //console.log("true");
28     return centinela;
29 }
```

Ahora se necesita identificar los motivos que se colocaron en el área de texto de la página web, mediante la función “crearmotivos” la variable se llama “motivos” y está enlazada al id=motivos (color verde, línea 32) que es el mismo id que está en la línea 86 del archivo “microsatelites.html”. Se declararon las variables “aux” y “aux2” compuestas por bases nitrogenadas que formarán arreglos. El código funciona como un bucle identificando los motivos del largo de nucleótidos especificado y usa el método push() para añadir ese largo de motivos al final de un arreglo devolviendo la nueva longitud del arreglo (a partir de la línea 40). Por ejemplo, si yo defino un tetranucleótido en el *input* y el *script* detecta el mismo tetranucleótido que se repite, lo va enlazando y generando un nuevo arreglo (de la línea 31 a la 54).

```
31  function crearmotivos() {
32      var motivos = Number.parseInt(document.getElementById("motivos").value);
33      //console.log(motivos);
34      var aux = new Array();
35      var aux2 = new Array();
36      var array = new Array();
37      aux = ["A", "T", "C", "G"];
38      aux2 = aux;
39      var a = 0;
40      for (var i = 0; i < motivos - 1; i++) {
41          array = new Array();
42          for (var p = 0; p < aux.length; p++) {
43              for (var j = 0; j < aux2.length; j++) {
44                  var valor = aux[p];
45                  //console.log(valor);
46                  //console.log(valor+aux2[j]);
47                  array.push(valor + aux2[j]);
48              }
49          }
50          aux = array;
51      }
52      // var repeticiones = Number.parseInt(document.getElementById("repeticiones").value);
53      return array;
54  }
```

JavaScript no dispone de constructores como tal, sino de llamadas de construcción a una función. “This” es una referencia que se crea cuando una función es invocada, no declarada. En este caso el siguiente código es para la tabla de resultados donde se muestra el motivo, donde inicia y termina dicho motivo en la cadena de ADN usando “this” (línea 55 a la 61).

```
55  class objetomotivo {
56      constructor(motivo, inicio, fin) {
57          this.motivo = motivo;
58          this.inicio = inicio;
59          this.fin = fin;
60      }
61  }
```

Para eliminar un elemento de un arreglo en JavaScript, se usó la función “removeItemFromArr”, por el método “indexOf() que retorna el primer índice en el que se puede encontrar un elemento dado en el arreglo y “splice” que cambia el contenido de un arreglo eliminando elementos existentes y/o agregando nuevos elementos.

```
63  function removeItemFromArr(arr, item) {
64      var i = arr.indexOf(item);
65      arr.splice(i, 1);
66  }
```

Se agregó la función “buscarcoincidencias” con la variable “motivos” y “cadena”. Por medio de “document.getElementById, se relaciona esta función con los id declarados en el documento “microsatelites.html” que son los que se encuentran en letra verde. El id “secuencia” de la línea 70 se describió en la línea 77 del HTML, “repeticiones” en la línea 92, “onclick” en la línea 97, etc. Este código busca las coincidencias de lo que se coloca en el formulario de la

página web y por medio de los identificadores declarados (id) o etiquetas, lo relaciona con este documento en JavaScript.

```
68  function buscarcoincidencias() {
69      var motivos = crearmotivos();
70      var cadena = document.getElementById("secuencia").value.toUpperCase();
71      cadena = cadena.replace(" ", "");
72      cadena = cadena.split("\n").join("");
73      var mot = Number.parseInt(document.getElementById("motivos").value);
74      var repeticiones = Number.parseInt(document.getElementById("repeticiones").value);
75      var tabla = document.getElementById("tabla_analisis");
76      var tbody = document.getElementById("tablainf");
```

Una vez que se validó la cadena, se genera un arreglo que contienen dicha cadena. El largo se especifica multiplicando el motivo por el número de repeticiones, de modo que si se busca tetranucleótidos repetidos 3 veces, se tiene una variable “fin” de 12 caracteres (4x3) (línea 82). Separamos la cadena de ADN en subcadenas según el tamaño del motivo, para después comparar el motivo con la subcadena (de la línea 84 a la 88). Se eliminan los pedazos menores al tamaño del motivo buscado (de la línea 90 a la 94).

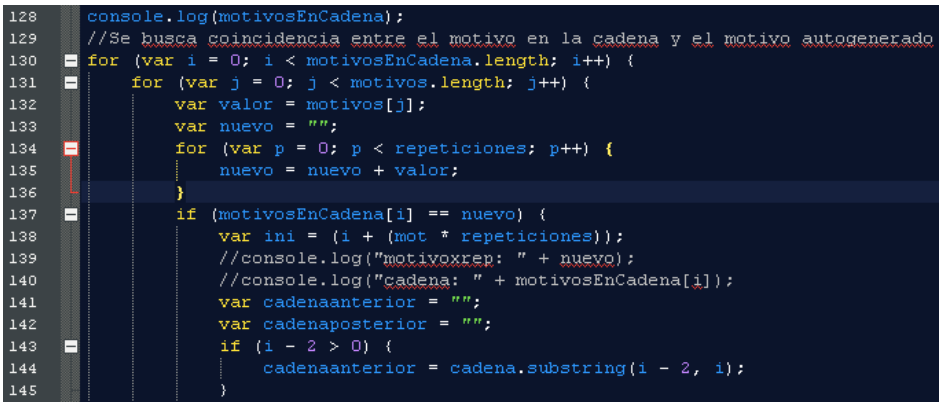
```
79  if(validarcadena(cadena)){
80      var array = new Array(); //Cadena
81      var ini = 0;
82      var fin = mot * repeticiones;
83      //Separamos la cadena en subcadenas según el tamai
84      for (var i = 0; i < cadena.length; i++) {
85          array.push(cadena.substring(ini, fin));
86          ini++;
87          fin = ini + (mot * repeticiones);
88      }
89      //Eliminar los pedazos menores al tamaño del motiv
90      for (var p = 0; p < array.length; p++) {
91          if (array[p].length < mot) {
92              // removeItemFromArr(array, array[p]);
93          }
94      }
```

Elimino los duplicados.

```
110     var motivosEnCadena = array; //[...new Set(array)];
111     var coincidencias = new Array();
112     var inciomotivo = 0,
113         finmotivo = 0;
114     console.log(motivosEnCadena);
115     for (var i = 0; i < motivosEnCadena.length; i++) {
116         var anterior = "";
117         var siguiente = "";
118         if (i - 1 > 0) {
119             anterior = motivosEnCadena[i - 1];
120         }
121         if (i + 1 < motivosEnCadena.length) {
122             siguiente = motivosEnCadena[i + 1];
123         }
124         if (anterior == motivosEnCadena[i] || siguiente == motivosEnCadena[i]) {
125             removeItemFromArr(motivosEnCadena, motivosEnCadena[i]);
126         }
127     }
```

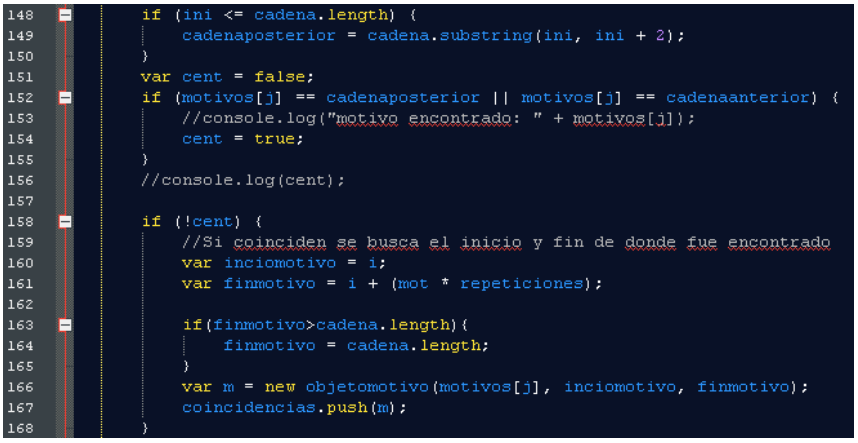
Se busca coincidencia entre el motivo en la cadena de ADN y el motivo autogenerado (de la línea 130 a la 145). En el ejemplo dado, si son

tetranucleótidos (motivo de 4), se busca la coincidencia con la subcadena siguiente (cadena posterior), se hace lo mismo hasta alcanzar el número de repeticiones del *input*.



```
128     console.log(motivosEnCadena);
129     //Se busca coincidencia entre el motivo en la cadena y el motivo autogenerado
130     for (var i = 0; i < motivosEnCadena.length; i++) {
131         for (var j = 0; j < motivos.length; j++) {
132             var valor = motivos[j];
133             var nuevo = "";
134             for (var p = 0; p < repeticiones; p++) {
135                 nuevo = nuevo + valor;
136             }
137             if (motivosEnCadena[i] == nuevo) {
138                 var ini = (i + (mot * repeticiones));
139                 //console.log("motivoxrep: " + nuevo);
140                 //console.log("cadena: " + motivosEnCadena[i]);
141                 var cadenaanterior = "";
142                 var cadenaposterior = "";
143                 if (i - 2 > 0) {
144                     cadenaanterior = cadena.substring(i - 2, i);
145                 }
```

Si coinciden se busca el inicio y final de donde este motivo fue encontrado (de la línea 160 a la 165).



```
148         if (ini <= cadena.length) {
149             cadenaposterior = cadena.substring(ini, ini + 2);
150         }
151         var cent = false;
152         if (motivos[j] == cadenaposterior || motivos[j] == cadenaanterior) {
153             //console.log("motivo encontrado: " + motivos[j]);
154             cent = true;
155         }
156         //console.log(cent);
157
158         if (!cent) {
159             //Si coinciden se busca el inicio y fin de donde fue encontrado
160             var inciomotivo = i;
161             var finmotivo = i + (mot * repeticiones);
162
163             if(finmotivo>cadena.length){
164                 finmotivo = cadena.length;
165             }
166             var m = new objetomotivo(motivos[j], inciomotivo, finmotivo);
167             coincidencias.push(m);
168         }
```

Si hay coincidencia entre el largo del motivo y el número de repeticiones solicitadas con la secuencia de ADN, se muestran los resultados en una tabla. Si alguna letra no coincide con las declaradas (A, G, C, T), se muestra que la cadena tiene letras no válidas (de la línea 172 a la 203).

```
172     console.log(coincidencias);
173     while (tbody.hasChildNodes()) {
174         tbody.removeChild(tbody.lastChild);
175     }
176     for (var p = 0; p < coincidencias.length; p++) {
177         console.log(coincidencias[p].motivo);
178         var fila = document.createElement("tr");
179         var celda = document.createElement("td");
180         var textoCelda = document.createTextNode(coincidencias[p].motivo);
181         celda.appendChild(textoCelda);
182         fila.appendChild(celda);
183         var celda = document.createElement("td");
184         var textoCelda = document.createTextNode(repeticiones);
185         celda.appendChild(textoCelda);
186         fila.appendChild(celda);
187         var celda = document.createElement("td");
188         var textoCelda = document.createTextNode(coincidencias[p].inicio );
189         celda.appendChild(textoCelda);
190         fila.appendChild(celda);
191         var celda = document.createElement("td");
192         var textoCelda = document.createTextNode(coincidencias[p].fin);
193         celda.appendChild(textoCelda);
194         fila.appendChild(celda);
195         var celda = document.createElement("td");
196         var textoCelda = document.createTextNode(cadena.length);
197         celda.appendChild(textoCelda);
198         fila.appendChild(celda);
199         tbody.appendChild(fila);
200     }
201     tabla.appendChild(tbody);
202 }else{
203     alert("La cadena contiene una letra no valida");
```

Se valida que los motivos van desde 2 hasta 10, tal como figura en el documento HTML (línea 209).

```
207 function llenarmotivos() {
208     var parent = document.getElementById("motivos");
209     for (var i = 2; i <= 10; i++) {
210         var child = document.createElement("option");
211         child.innerHTML = "" + i;
212         var last = parent.lastChild;
213         parent.insertBefore(child, last);
214     }
215 }
```

ANEXO 6.4: Código (HTML, CSS y Javascript) usado para el desarrollo de la herramienta que determina si el vector está en fase.

6.4.1. “fase.html”

En el documento “fase.html”, se coloca el código de lo que se verá en la web y se lo enlaza al documento escrito en JavaScript “analisistripletes.js” que contiene el código fuente que separa la cadena de ADN cada tres bases nitrogenadas (línea 66). Como se puede apreciar, una vez más se coloca un formulario, para que en un área de texto, se pueda ingresar la secuencia de ADN a analizar (línea 73-74) y la identificamos con la etiqueta “id=secuencia (línea 74).

```
66    <script type="text/javascript" src="assets/js/analisistripletes.js"></script>
67  </head>
68  <body>
69      <div class="container">
70      <div class="flex-row">
71        <form id="formulario">
72          <div class="form-group"><font face="kefa" color="black">
73            <label for="secuencia_adn">Enter a DNA sequence</label>
74            <textarea style="height: 10em;" name="secuencia_adn" id="secuencia" class="form-control" cols="30" r
75              required></textarea>
76          </div>
77          <div class="d-flex justify-content-end"><font face="kefa" color="black">
78            <button type="button" class="btn btn-dark" id="buscar" onclick="numerodetripletes()">Search</button>
79          </div>
80        </form>
```

Por otro lado, colocamos un “card-title” y un “card-text” que son herramientas en bootstrap, una biblioteca multiplataforma de código abierto para diseño de sitios web, para que se muestre un rectángulo oscuro donde se verán el número de tripletes de la cadena (línea 88-89), si se encuentran en fase o no (línea 96) y un área de resultados, donde se mostrará a la cadena de ADN, separada de tres en tres (línea 100).

```
82      <div class="flex-row">
83        <p class="text-break"></p>
84      </div>
85      <div class="flex-row">
86        <div class="card text-white bg-dark mb-3">
87          <div class="card-body text-center">
88            <h3 class="card-title" id="tripletes"></h3>
89            <h5 class="card-text" id="nombre_etiqueta"></h5>
90          </div>
91        </div>
92      </div>
93      <div class="flex-row">
94        <div class="card text-white bg-dark mb-3">
95          <div class="card-body text-center">
96            <h5 class="card-title" id="fase"></h5>
97          </div>
98        </div>
99      </div>
100     <div class="flex-row" id="cadena_separada">
```

6.4.2. “analisistripletes.js”

El archivo “analisistripletes.js” contiene la programación *per se*. Para ello, primero se realiza la validación de la cadena de ADN ingresada en el área de texto por medio de la función “validarcadena”, se define la variable cadena y ese valor se divide por medio del método split() (líneas 2 a la 4). Se valida que la cadena de ADN no presente números, pues solo debe presentar las letras correspondientes a las bases nitrogenadas (línea de la 6 a la 10).

```
2  function validarcadena(valor) {
3      var cadena = valor.split("");
4      var centinela = true;
5  //valido que esta no contenga números
6      cadena.forEach(element => {
7          element = element.toLowerCase();
8          if (!isNaN(Number.parseInt(element, 10))) {
9              //console.log("false");
10             centinela = false;
11         }
12     });
```

Se especifica que las letras que se reconozcan sean solo A, T, C y G (de la línea 13 a la 22).

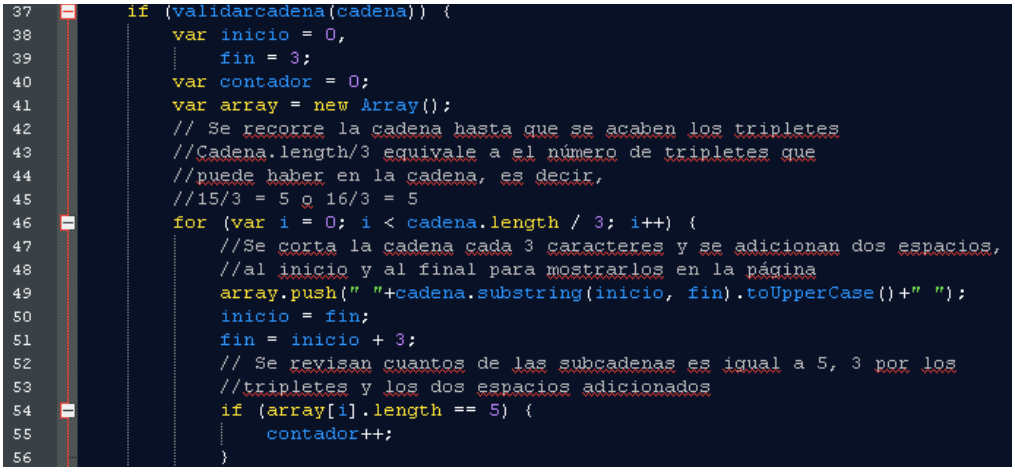
```
14     cadena.forEach(element => {
15         element = element.toUpperCase();
16         if (element != "A") {
17             if (element != "G") {
18                 if (element != "C") {
19                     if (element != "T") {
20                         centinela = false;
21                     }
22                 }
```

Con “return centinela;” se indica que si las validaciones están bien, es decir, si cumple que el *input* ingresado son letras y que son las especificadas, se retorne “verdadero”, caso contrario retorne “falso”, con lo cual no se ejecutará el *script* (línea 27). Para verificar el número de tripletes que hay en la cadena se usa la función “numerodetripletes” y la variable “cadena” (línea 30 y 31), relacionando este archivo con el de “fase.html” mediante la etiqueta “secuencia”. Se obtiene la cadena ingresada que es la que analiza el *script* (línea 32). En un inicio se consideró que la secuencia de ADN termina cuando hay un espacio, pero eso no se cumplía para las secuencias FASTA, que tienen en cada renglón 70 bases nitrogenadas. Para ello, se deben eliminar los espacios en el caso de las secuencias FASTA para que se puedan procesar, eso se realiza con reemplazar el espacio con la función “replace”, separar con “split” y luego unir con “join”, de forma que en lugar de ser varias líneas de 70 caracteres, sean una sola cadena procesable (línea 34 y 35).

```
27     return centinela;
28 }
29 //Verifica el número de tripletes existentes en la cadena
30 function numerodetripletes() {
31     var cadena = "";
32     cadena = document.getElementById("secuencia").value;//Se obtiene la cadena
33     //Se eliminan los espacios en el caso de las cadenas en formato FASTA
34     cadena = cadena.replace(" ", "");
35     cadena = cadena.split("\n").join("");
```

La lógica de la programación es dividir la cadena de ADN ingresada de tres en tres, de modo que se forman subcadenas de tres caracteres (*strings*

que se unen en arreglos), se verifica que la cadena sea correcta (de la línea 37 a la 41). El *script* recorre la cadena hasta que se acaben los tripletes, los cuenta por medio de la variable “contador” (línea 40) y reconoce el número de tripletes que hay en la cadena ingresada por medio de cadena.length/3, justamente está dividido entre tres por ser múltiplo de ese número (línea 46). Para poder mostrar los tripletes en la página web es necesario cortar la cadena cada tres caracteres, agregar dos espacios, uno al inicio del triplete y otro al final (de la línea 49 a la 51), revisar cuantas de las subcadenas es igual a cinco (tres caracteres de los tripletes y dos de los espacios adicionados) (línea 54 y 55).



```
37      if (validarcadena(cadena)) {
38          var inicio = 0,
39              fin = 3;
40          var contador = 0;
41          var array = new Array();
42          // Se recorre la cadena hasta que se acaben los tripletes
43          //Cadena.length/3 equivale a el número de tripletes que
44          //puede haber en la cadena, es decir,
45          //15/3 = 5 o 16/3 = 5
46          for (var i = 0; i < cadena.length / 3; i++) {
47              //Se corta la cadena cada 3 caracteres y se adicionan dos espacios,
48              //al inicio y al final para mostrarlos en la página
49              array.push(" "+cadena.substring(inicio, fin).toUpperCase()+" ");
50              inicio = fin;
51              fin = inicio + 3;
52              // Se revisan cuantos de las subcadenas es igual a 5, 3 por los
53              //tripletes y los dos espacios adicionados
54              if (array[i].length == 5) {
55                  contador++;
56              }
```

Por un lado, se contó los tripletes con el contador, por otro se los separó y se les incluyó espacios a ambos lados, si el número dado por el contador es igual al número de caracteres cortados, quiere decir que la secuencia está en fase (es múltiplo de 3). Si estos números no coinciden es porque la secuencia dada no está en fase (de la línea 60 a la 65). Por último se agregan las respuestas en las respectivas etiquetas identificadas en el documento “fase.html” para que se pueda ver en la página web (de la línea 67 a la 70). Adicionalmente, si hubiera algún número, signo o letra que no corresponde al ADN saldrá que la secuencia no puede ser analizada (línea 71 y 72)

```
60          var respuesta = "";
61          if (contador == array.length) {
62              respuesta = "Sequence in Phase";
63          } else {
64              respuesta = "Sequence is not in Phase";
65          }
66          // Se agregan las respuestas en las respectivas etiquetas
67          document.getElementById("tripletes").innerHTML = contador;
68          document.getElementById("nombre_etiqueta").innerHTML = "Numbers of triplets";
69          document.getElementById("fase").innerHTML = respuesta;
70          document.getElementById("cadena_separada").innerHTML= "<br/>The separate DNA sequence is: <br/> " + array;
71      } else {
72          document.getElementById("tripletes").innerHTML = "The sequence can not be analyzed, you should verify that
73      }
```

ANEXO 6.5: Código (HTML, CSS y Javascript) usado para el desarrollo de la herramienta que busca sitios de restricción.

6.5.1. "restricción.html"

Se creó el archivo "restricción.html" que usa los documentos en JavaScript "enzimas.js" que es una base de datos que contiene el nombre de todas las enzimas de restricción con los cortes especificados (línea 65) y "analisisenzimas.js" que tiene la lógica de programación para reconocer qué enzima y donde realiza el corte (línea 66).

```
65      <script type="text/javascript" src="assets/js/enzimas.js"></script>
66      <script type="text/javascript" src="assets/js/analisisenzimas.js"></script>
67      </head>
68      <body onload="obtenerCoincidencias()">
69      <div class="container">
```

En el documento HTML se colocó a manera de formulario, un área de texto para ingresar la secuencia de ADN (línea 73) y se identificó la etiqueta "secuencia" (línea 74).

```
70      <div class="flex-row">
71          <form class="" id="formulario">
72              <div class="form-group">
73                  <label for="secuencia_adn">Enter a DNA sequence</label>
74                  <textarea style="height: 10em;" name="secuencia_adn" id="secuencia" class="form-control" cols="30"
75                      rows="10" required></textarea>
76              </div>
77                  <button type="button" class="btn btn-primary" id="buscar" onclick="obtenerCoincidencias()">Search</button>
78          </form>
79      </div>
```

Por último se colocó una tabla resultados que contienen el número de enzima, el nombre de la misma, la secuencia que hay antes del corte, la secuencia después del corte y la posición en la que realiza la escisión (de la línea 80 a la 97).

```
80          <div class="flex-row">
81              <div class="table-responsive ">
82              <table class="table" id="tabla_analisis">
83                  <thead class="thead-dark">
84                      <tr>
85                          <th scope="col">#</th>
86                          <th scope="col">Enzyme</th>
87                          <th scope="col">Before</th>
88                          <th scope="col">After</th>
89                          <th scope="col">Cleavage position</th>
90                      </tr>
91                  </thead>
92
93                  <tbody id="tablainf">
94
95                  </tbody>
96
97              </table>
```

6.5.2. “enzimas.js”

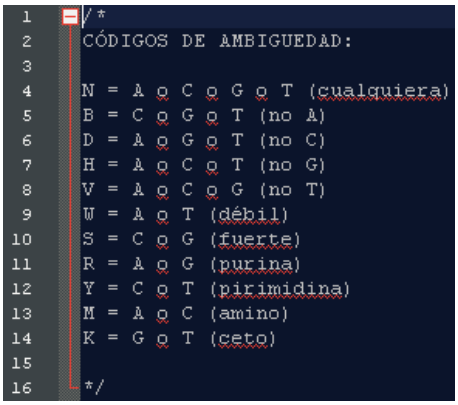
El archivo “enzimas.js” es una base de datos donde se colocaron las 286 enzimas disponibles en la web de NEB (ver materiales y métodos), por medio de la variable “enzimas” la cual será usada en los archivos JavaScript. Se especificó por ejemplo el nombre EcoRI (línea 144) que como sabemos reconoce el motivo G|AATTC y realiza la escisión luego de G (donde está la barra vertical roja), por ello la cadena “antes” del corte es G y la cadena “después” del corte es AATTC, eso se hizo para todas las enzimas.

```
var enzimas = [{
        "nombre": "BsmAI",
        "antes": "GTCTCN",
        "despues": "N"
    }, {
        "nombre": "BsmFI",
        "antes": "GGGACNNNNNNNNNN",
        "despues": "NNNN"
    }, {
        "nombre": "BsmI",
        "antes": "GAATGCN",
        "despues": "N"
    }, {
```

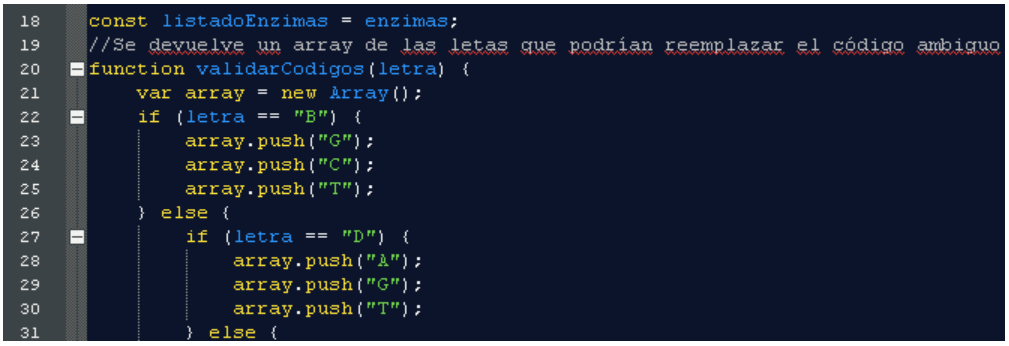
```
144             "nombre": "EcoRI",
145             "antes": "G",
146             "despues": "AATTC"
147         }, {
```

6.5.3. “analisisenzima.js”

El archivo “analisisenzima.js” contiene la programación para poder realizar el análisis de restricción, para ello debemos tener claro los códigos de ambigüedad especificados en la base de datos de NEB Tools (línea 4 a la 14).



Se usa la función “const”, la cual es igual que “let” pero no permite reasignar su valor, lo que permite minimizar el estado de mutación, ya que el código estará corriendo muchos procesos a la vez, de esta forma se declara el listado de enzimas (línea 18) disponible en “enzimas.js”. Luego, se crea un array de las letras que podrían reemplazar el código ambiguo con la función “validarCodigos”, por ejemplo, si la enzima tiene una base “B”, significa que puede ser G, C o T (de la línea 22 a la 26) y se hizo lo mismo para cada letra ambigua que podemos encontrar (a partir de la línea 22 hasta la 74, la cual no se muestra).



Para obtener las enzimas que cortan la cadena de ADN y la cadena antes del corte, después del corte y el lugar de clivaje se usa la función “obtenerCoincidencias”, las cuales están enlazadas a la secuencia de ADN que se coloca en la página web, representada mediante la etiqueta “secuencia” (línea 82) y que se declaró en el archivo “restricción.html”. Se eliminan los espacios, para poder aceptar las cadenas FASTA, mediante el reemplazo del espacio en blanco, una separación de la subcadena y la posterior unión (de la línea 84 a la 86).

```
75          }
76      }
77      return array;
78  }
79  let coincidencias = [];
80  //Se obtienen las enzimas que cortan la cadena,su antes, su
81  function obtenerCoincidencias() {
82      var valor = document.getElementById("secuencia").value;
83      //Se eliminan los espacios para aceptar cadenas tipo FA
84      valor = valor.replace(" ", "");
85      valor = valor.split("\n").join("");
86      var cadena = valor.split("");
```

Se recorre cada base nitrogenada a lo largo de la cadena de ADN (línea 89), buscando el motivo anterior al corte que realiza cada una de las enzimas de restricción del listado (línea 90), por ejemplo en el caso de EcoRI (G|AATTC) sería hasta G (antes de la barra vertical).

```
88      for (var i = 0; i < cadena.length; i++) {
89          //Se recorre el listado de enzimas
90          for (var j = 0; j < listadoEnzimas.length; j++) {
91              var cent = false;
92              var antes = [];
```

Si la secuencia antes del corte es igual a una sola letra (ejemplo: EcoRI), se toma esa letra. Si tiene más bases nitrogenadas antes de la escisión (por ejemplo: AfeI, realiza el corte AGC|GCT), se toma la primera letra de la secuencia (línea 96 a la 101), es decir, en el caso de AfeI solo toma la letra A.

```
96              if (listadoEnzimas[j].antes.length > 1) {
97                  antes = listadoEnzimas[j].antes.split('');
98              } else {
99                  antes = listadoEnzimas[j].antes;
100             }
101             var despues = [];
```

Si la secuencia después del corte es igual a una sola letra (por ejemplo Nb.BsmI, realiza la escisión GAATG|C), se toma esa letra (en el ejemplo se tomaría C), si tiene más bases nitrogenadas luego del corte, se obtiene la primera letra de la secuencia (en el caso de EcoRI, con escisión G|AATTC, se tomaría la primera Adenina).

```
105         if (listadoEnzimas[j].despues.length > 1) {
106             despues = listadoEnzimas[j].despues.split('');
107         } else {
108             despues = listadoEnzimas[j].despues;
109         }
```

Se obtiene una subcadena del tamaño de la secuencia anterior al corte de la enzima (línea 111), por ejemplo Nb.BsmI, que realiza la escisión GAATG|C, la subcadena es de un largo de cinco letras. Se valida que la secuencia antes del corte sea igual a la subcadena anterior (línea 113). Se

ubica la posición luego de la escisión (línea 115). Se obtiene una subcadena con el tamaño de la secuencia posterior al corte de la enzima, desde la posición de la escisión (línea 117). En el caso de EcoRI, con escisión G|AATTC, se buscaría una subcadena de largo cinco, luego se valida que la subcadena luego del corte tenga la misma secuencia (línea 119).

```
111         var c = valor.substring(i, i + antes.length).split('');
112         //Se valida que la secuencia antes de corte de la cadena sea
113         var contantes = validarCorte(antes, c);
114         //posición después del corte
115         var iniD = i + antes.length;
116         //Se obtiene una subcadena con el tamaño de la secuencia pos
117         c = valor.substring(iniD, iniD + despues.length).split('');
118         //Se valida dque la secuencia después del corte sea igual a
119         var contdespues = validarCorte(despues, c);
```

Se coloca la condicional que si ambas subcadenas son idénticas al corte que deberían producir cada una de ellas, entonces la enzima corta a la cadena en ese punto (de la línea 121 a la 130). Si la enzima corta la cadena, se guarda la enzima y la posición de corte en un arreglo de coincidencias (de la línea 132 a la 137).

```
121     if (contantes == antes.length && contdespues == despues.length) {
122         cent = true;
123     } else {
124         cent = false;
125     }
126     if (cent == true) {
127         var coincidencia = {
128             "enzima": [],
129             "corte": 0
130         }
131         // Si la enzima corta la cadena se guarda la enzima y la posic
132         coincidencia.enzima = listadoEnzimas[j];
133         coincidencia.corte = iniD;
134         coincidencias.push(coincidencia);
135         // console.log("enzimas de corte: " + coincidencias);
136     }
137 }
```

Se eliminan los valores de la tabla si no coinciden.

```
146     var tabla = document.getElementById("tabla_analisis");
147     var tbody = document.getElementById("tablainf");
148     while (tbody.hasChildNodes()) {
149         tbody.removeChild(tbody.lastChild);
150     }
```

Se lista la información de las enzimas coincidentes en la tabla (líneas 152-183).

```
152     for (var p = 0; p < coincidencias.length; p++) {
153         // coincidencias[p].enzima.antes = valor.substring(0, coincidencias[p].cor
154         // coincidencias[p].enzima.despues = valor.substring(coincidencias[p].cort
155         var fila = document.createElement("tr");
156         var celda = document.createElement("td");
157         var textoCelda = document.createTextNode(p + 1);
158         celda.appendChild(textoCelda);
159         fila.appendChild(celda);
160         var celda = document.createElement("td");
161         var textoCelda = document.createTextNode(coincidencias[p].enzima.nombre);
162         celda.appendChild(textoCelda);
163         fila.appendChild(celda);
```

```
181     }
182     tabla.appendChild(tbody);
183 }
```

Se usa la función que valida que cada cadena sea igual a la secuencia de corte de la enzima, tanto la anterior como la posterior (líneas 186-203).

```
186 function validarCorte(corte, cadena) {
187     var contador = 0;
188     for (var i = 0; i < cadena.length; i++) {
189         var arrayCodigoAmbiguo = [];
190         if (cadena[i] == corte[i]) {
191             contador++;
192         } else {
193             if (corte[i] == "N") {
194                 contador++;
195             } else {
196                 //console.log("false");
197                 arrayCodigoAmbiguo = validarCodigos(corte[i]);
198                 for (var t = 0; t < arrayCodigoAmbiguo.length; t++) {
199                     if (cadena[i] == arrayCodigoAmbiguo[t]) {
200                         contador++;
201                     }
202                 }
203             }
204         }
205     }
206     return contador;
207 }
```

ANEXO 6.6: Código (SQL) usado para la construcción de la base de datos.

A continuación, se muestran las tablas en PHPmyadmin y la sentencias SQL que se usaron para la creación de las mismas, si bien se incluyen las FKs en las tablas, su asignación solo se da cuando se establecen las relaciones:

```
-- Estructura de tabla para la tabla `Proteins`
--

CREATE TABLE `Proteins` (
  `IDprot` varchar(10) COLLATE latin1_spanish_ci NOT NULL,
  `IDuniprot` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `GI` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `id_name` varchar(15) COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	**IDprot**	varchar(10)	latin1_spanish_ci		No	*Ninguna*
☐	2	**IDuniprot**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	3	**GI**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	4	**id_name**	varchar(15)	latin1_spanish_ci		No	*Ninguna*

Figura 6.6.1. Tabla Proteins en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
-- Estructura de tabla para la tabla `ProteinChar`
--

CREATE TABLE `ProteinChar` (
  `id_prot` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `aanumber` int(10) NOT NULL,
  `fastaprot` text COLLATE latin1_spanish_ci NOT NULL,
  `weight` float(10,3) NOT NULL,
  `pI` float(5,2) NOT NULL,
  `instindex` float(6,2) NOT NULL,
  `alipindex` float(6,2) NOT NULL,
  `GRAVY` float(7,3) NOT NULL,
  `nameprot` varchar(15) COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	**id_prot**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	2	**aanumber**	int(10)			No	*Ninguna*
☐	3	**fastaprot**	text	latin1_spanish_ci		No	*Ninguna*
☐	4	**weight**	float(10,3)			No	*Ninguna*
☐	5	**pI**	float(5,2)			No	*Ninguna*
☐	6	**instindex**	float(6,2)			No	*Ninguna*
☐	7	**alipindex**	float(6,2)			No	*Ninguna*
☐	8	**GRAVY**	float(7,3)			No	*Ninguna*
☐	9	**nameprot**	varchar(15)	latin1_spanish_ci		No	*Ninguna*

Figura 6.6.2. Tabla ProteinsChar en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
-- Estructura de tabla para la tabla `Protname`
--

CREATE TABLE `Protname` (
  `IDname` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `name1` varchar(20) COLLATE latin1_spanish_ci DEFAULT NULL,
  `name2` varchar(20) COLLATE latin1_spanish_ci DEFAULT NULL,
  `name3` varchar(20) COLLATE latin1_spanish_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	**IDname** 🔑	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	2	**name1**	varchar(20)	latin1_spanish_ci		Sí	*NULL*
☐	3	**name2**	varchar(20)	latin1_spanish_ci		Sí	*NULL*
☐	4	**name3**	varchar(20)	latin1_spanish_ci		Sí	*NULL*

Figura 6.6.3. Tabla Protname en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
-- Estructura de tabla para la tabla `Species`
--

CREATE TABLE `Species` (
  `IDsp` varchar(10) COLLATE latin1_spanish_ci NOT NULL,
  `namesp` varchar(30) COLLATE latin1_spanish_ci NOT NULL,
  `gender` varchar(30) COLLATE latin1_spanish_ci NOT NULL,
  `sp` varchar(30) COLLATE latin1_spanish_ci NOT NULL,
  `idaf` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL,
  `IdGene` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL,
  `id_pro` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL,
  `idbf` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	**IDsp** 🔑	varchar(10)	latin1_spanish_ci		No	*Ninguna*
☐	2	**namesp**	varchar(30)	latin1_spanish_ci		No	*Ninguna*
☐	3	**gender**	varchar(30)	latin1_spanish_ci		No	*Ninguna*
☐	4	**sp**	varchar(30)	latin1_spanish_ci		No	*Ninguna*
☐	5	**idaf** 🔑	varchar(15)	latin1_spanish_ci		Sí	*NULL*
☐	6	**IdGene** 🔑	varchar(15)	latin1_spanish_ci		Sí	*NULL*
☐	7	**id_pro** 🔑	varchar(15)	latin1_spanish_ci		Sí	*NULL*
☐	8	**idbf** 🔑	varchar(15)	latin1_spanish_ci		Sí	*NULL*

Figura 6.6.4. Tabla Species en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
-- Estructura de tabla para la tabla `IndSyst`
--

CREATE TABLE `IndSyst` (
  `IDsys` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `type` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `idprot` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `id_sp` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `id_af` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL,
  `Idpap` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL,
  `IDGene` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL,
  `id_bf` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	IDsys	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	2	type	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	3	idprot	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	4	id_sp	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	5	id_af	varchar(15)	latin1_spanish_ci		Sí	*NULL*
☐	6	idpap	varchar(15)	latin1_spanish_ci		Sí	*NULL*
☐	7	IDGene	varchar(15)	latin1_spanish_ci		Sí	*NULL*
☐	8	id_bf	varchar(15)	latin1_spanish_ci		Sí	*NULL*

Figura 6.6.5. Tabla IndSyst en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
-- Estructura de tabla para la tabla `Gene`
--

CREATE TABLE `Gene` (
  `idgen` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `IDNCBI` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `idname` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `id_prot` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `id_char` varchar(15) COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	idgen	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	2	IDNCBI	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	3	idname	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	4	id_prot	varchar(15)	latin1_spanish_ci		No	*Ninguna*
☐	5	id_char	varchar(15)	latin1_spanish_ci		No	*Ninguna*

Figura 6.6.6. Tabla Gene en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
-- Estructura de tabla para la tabla `Genname`
--

CREATE TABLE `Genname` (
  `ID_name` varchar(10) COLLATE latin1_spanish_ci NOT NULL,
  `name1` varchar(20) COLLATE latin1_spanish_ci DEFAULT NULL,
  `name2` varchar(20) COLLATE latin1_spanish_ci DEFAULT NULL,
  `name3` varchar(20) COLLATE latin1_spanish_ci DEFAULT NULL,
  `idcchar` varchar(10) COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	ID_name	varchar(10)	latin1_spanish_ci		No	*Ninguna*
☐	2	name1	varchar(20)	latin1_spanish_ci		Sí	*NULL*
☐	3	name2	varchar(20)	latin1_spanish_ci		Sí	*NULL*
☐	4	name3	varchar(20)	latin1_spanish_ci		Sí	*NULL*
☐	5	idcchar	varchar(10)	latin1_spanish_ci		No	*Ninguna*

Figura 6.6.7. Tabla Genname en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
-- Estructura de tabla para la tabla `GenChar`
--

CREATE TABLE `GenChar` (
  `ID_char` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `namegen` varchar(20) COLLATE latin1_spanish_ci NOT NULL,
  `fastagen` text COLLATE latin1_spanish_ci NOT NULL,
  `function` text COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**ID_char**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
2	**namegen**	varchar(20)	latin1_spanish_ci		No	*Ninguna*
3	**fastagen**	text	latin1_spanish_ci		No	*Ninguna*
4	**function**	text	latin1_spanish_ci		No	*Ninguna*

Figura 6.6.8. Tabla GenChar en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
--
-- Estructura de tabla para la tabla `BioticF`
--

CREATE TABLE `BioticF` (
  `IDpat` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `PatGenus` varchar(30) COLLATE latin1_spanish_ci NOT NULL,
  `Patspecie` varchar(30) COLLATE latin1_spanish_ci NOT NULL,
  `Pattype` varchar(30) COLLATE latin1_spanish_ci NOT NULL,
  `idgene` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `id_phe` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `id_paper` varchar(15) COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**IDpat**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
2	**PatGenus**	varchar(30)	latin1_spanish_ci		No	*Ninguna*
3	**Patspecie**	varchar(30)	latin1_spanish_ci		No	*Ninguna*
4	**Pattype**	varchar(30)	latin1_spanish_ci		No	*Ninguna*
5	**idgene**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
6	**id_phe**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
7	**id_paper**	varchar(15)	latin1_spanish_ci		No	*Ninguna*

Figura 6.6.9. Tabla BioticF en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
-- Estructura de tabla para la tabla `AbioticF`
--

CREATE TABLE `AbioticF` (
  `IDstress` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `stress` text COLLATE latin1_spanish_ci NOT NULL,
  `idgene` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `id_phe` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `id_paper` varchar(15) COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**IDstress**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
2	**stress**	text	latin1_spanish_ci		No	*Ninguna*
3	**idgene**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
4	**id_phe**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
5	**id_paper**	varchar(15)	latin1_spanish_ci		No	*Ninguna*

Figura 6.6.10. Tabla AbioticF en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

```
-- Estructura de tabla para la tabla `Phenotype`
--

CREATE TABLE `Phenotype` (
  `IDphenot` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `pheno1` text COLLATE latin1_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**IDphenot**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
2	**pheno1**	text	latin1_spanish_ci		No	*Ninguna*

Figura 6.6.11. Tabla Phenotype en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

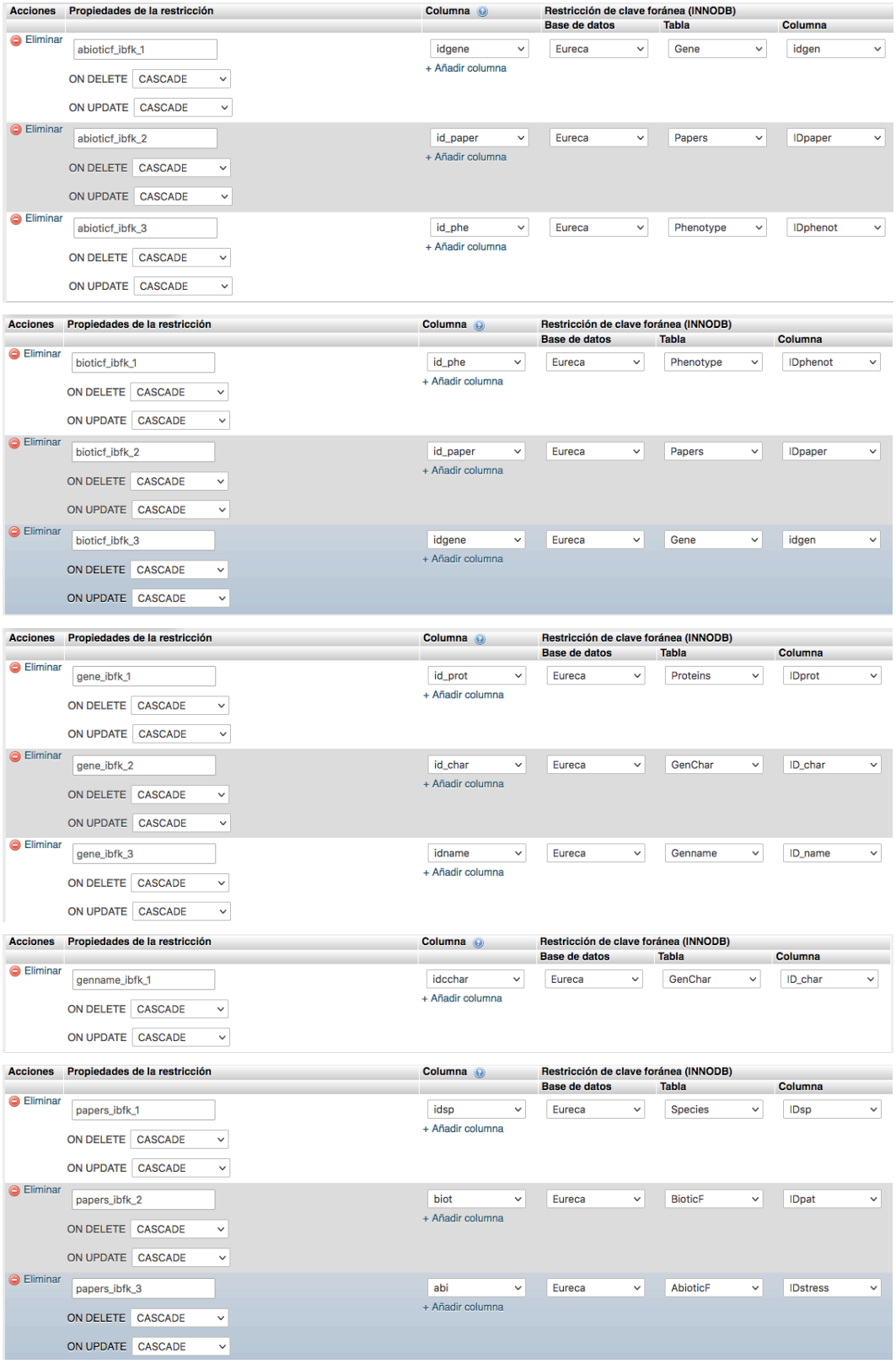
```
-- Estructura de tabla para la tabla `Papers`
--

CREATE TABLE `Papers` (
  `IDpaper` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `linkpaper` text COLLATE latin1_spanish_ci NOT NULL,
  `author` text COLLATE latin1_spanish_ci NOT NULL,
  `year` varchar(10) COLLATE latin1_spanish_ci NOT NULL,
  `idsp` varchar(15) COLLATE latin1_spanish_ci NOT NULL,
  `biot` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL,
  `abi` varchar(15) COLLATE latin1_spanish_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**IDpaper**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
2	**linkpaper**	text	latin1_spanish_ci		No	*Ninguna*
3	**author**	text	latin1_spanish_ci		No	*Ninguna*
4	**year**	varchar(10)	latin1_spanish_ci		No	*Ninguna*
5	**idsp**	varchar(15)	latin1_spanish_ci		No	*Ninguna*
6	**biot**	varchar(15)	latin1_spanish_ci		Sí	*NULL*
7	**abi**	varchar(15)	latin1_spanish_ci		Sí	*NULL*

Figura 6.6.12. Tabla Papers en PHPmyadmin con los tipos de datos identificados al igual que la PK (se incluyen las FKs). Se muestra la sentencia SQL (arriba).

Por último, se relacionaron las tablas y se agregó la integridad referencial para las 25 relaciones:



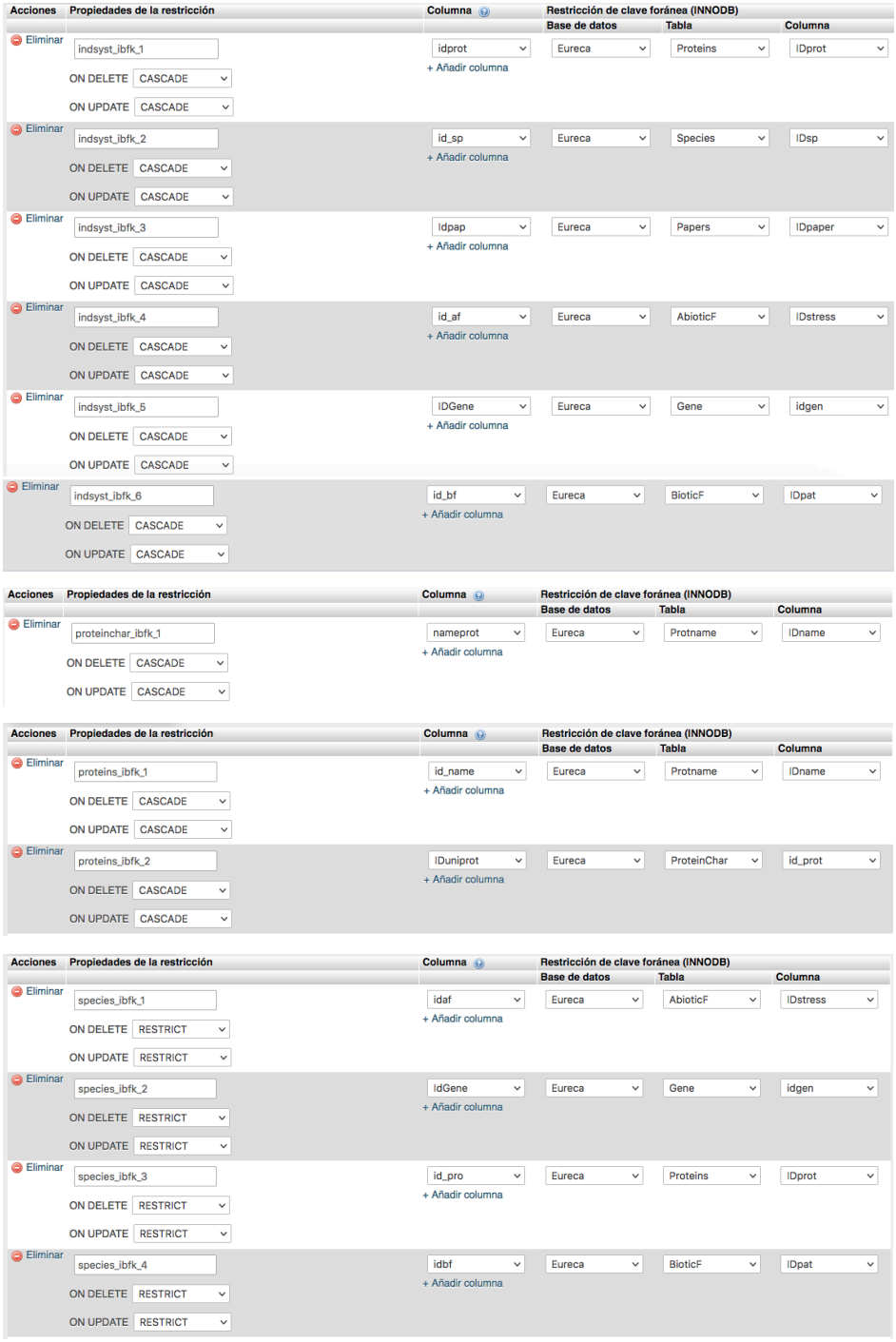
Acciones	Propiedades de la restricción	Columna	Restricción de clave foránea (INNODB)		
			Base de datos	Tabla	Columna
Eliminar	abioticf_ibfk_1	idgene	Eureca	Gene	idgen
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				
Eliminar	abioticf_ibfk_2	id_paper	Eureca	Papers	IDpaper
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				
Eliminar	abioticf_ibfk_3	id_phe	Eureca	Phenotype	IDphenot
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				

Acciones	Propiedades de la restricción	Columna	Restricción de clave foránea (INNODB)		
			Base de datos	Tabla	Columna
Eliminar	bioticf_ibfk_1	id_phe	Eureca	Phenotype	IDphenot
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				
Eliminar	bioticf_ibfk_2	id_paper	Eureca	Papers	IDpaper
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				
Eliminar	bioticf_ibfk_3	idgene	Eureca	Gene	idgen
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				

Acciones	Propiedades de la restricción	Columna	Restricción de clave foránea (INNODB)		
			Base de datos	Tabla	Columna
Eliminar	gene_ibfk_1	id_prot	Eureca	Proteins	IDprot
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				
Eliminar	gene_ibfk_2	id_char	Eureca	GenChar	ID_char
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				
Eliminar	gene_ibfk_3	idname	Eureca	Genname	ID_name
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				

Acciones	Propiedades de la restricción	Columna	Restricción de clave foránea (INNODB)		
			Base de datos	Tabla	Columna
Eliminar	genname_ibfk_1	idcchar	Eureca	GenChar	ID_char
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				

Acciones	Propiedades de la restricción	Columna	Restricción de clave foránea (INNODB)		
			Base de datos	Tabla	Columna
Eliminar	papers_ibfk_1	idsp	Eureca	Species	IDsp
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				
Eliminar	papers_ibfk_2	biot	Eureca	BioticF	IDpat
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				
Eliminar	papers_ibfk_3	abi	Eureca	AbioticF	IDstress
	ON DELETE CASCADE	+ Añadir columna			
	ON UPDATE CASCADE				



ANEXO 6.7: Modelo relacional de la base de datos en PHPmyadmin.

Al consultar el modelo relacional de la BD en la pestaña “Diseñador” en PHPmyadmin, se obtuvo lo siguiente (figura 6.7.1):

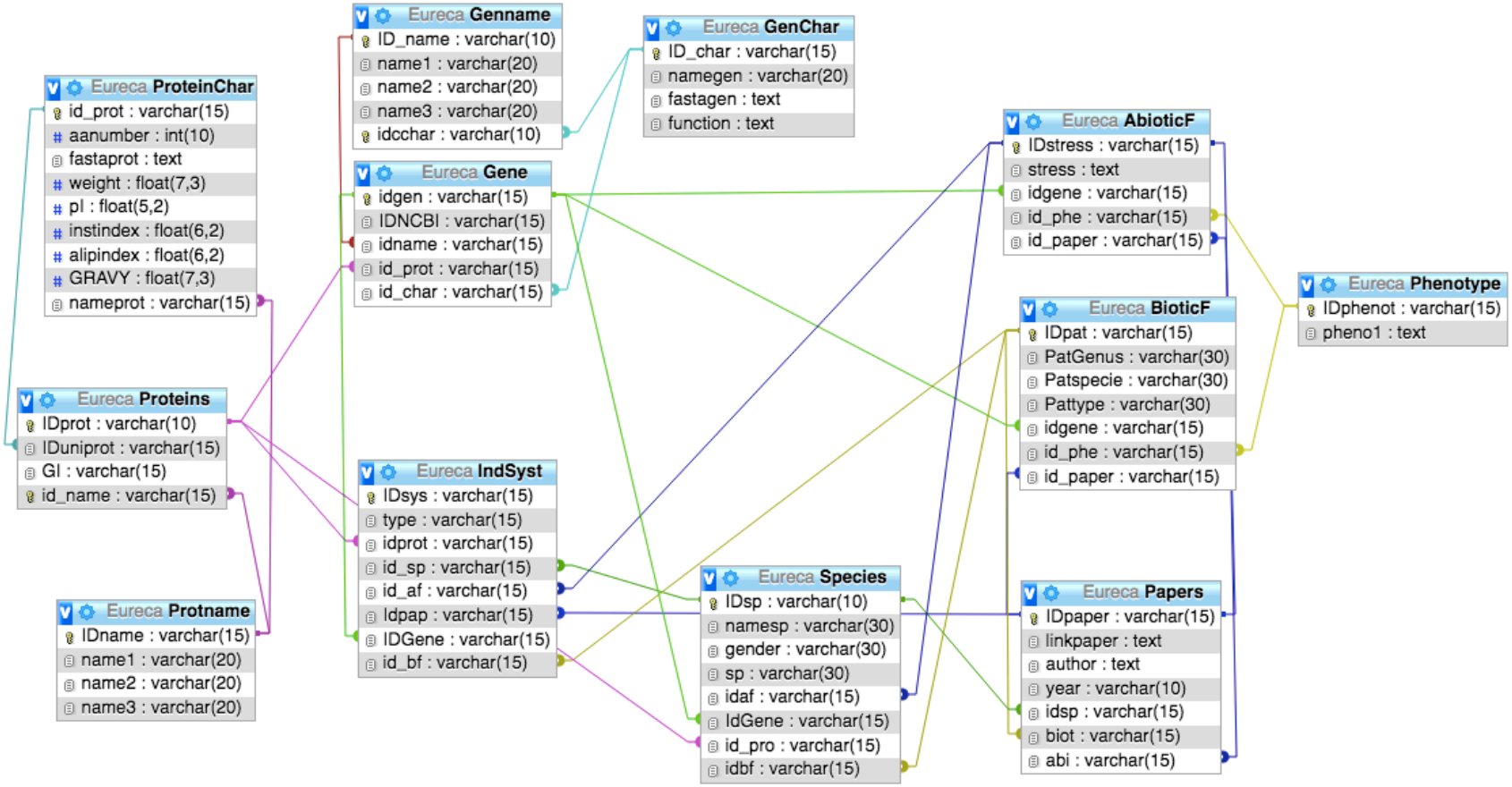


Figura 6.7.1. Modelo relacional en PHPmyadmin

ANEXO 6.8: Código (HTML, CSS y PHP) para la construcción y validación del formulario.

6.8.1. “validado_contacto.php”

Contiene el código HTML con el formulario dentro de él. El método elegido fue “POST” usado para enviar datos a un servidor y así crear o actualizar un recurso, los datos enviados al servidor con POST son almacenados en el requerimiento del cuerpo del HTTP. POST es también uno de los métodos HTTP más populares y a diferencia de los GET, éstos no permanecen en el historial del navegador, no pueden ser marcados como favoritos, no tienen restricción de largo de datos y son mucho más seguros.

A continuación, se muestra el código del archivo validado_contacto.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="estilo1.css">
    <title>Formulario de contacto de EURECA</title>
</head>
</body>
    <h1>Contact us </h1>
    <div class="contenedor">
        <form class="formulario" action="<?php htmlspecialchars ($_SERVER['PHP_SELF']); ?>" method="post">
            <label for="nombre" class="texto">Name <span>(required)</span></label>
            <input type="text" name="nombre" class="input" value="<?php if(!$enviar && isset($nombre)) echo $nombre ?>">
            <label for="apellido" class="texto">Last name <span>(required)</span></label>
            <input type="text" name="apellido" class="input" value="<?php if(!$enviar && isset($apellido)) echo $apellido ?>">
            <label for="correo" class="texto">E-mail <span>(required)</span></label>
            <input type="text" name="correo" class="input" value="<?php if(!$enviar && isset($correo)) echo $correo ?>">
                <label for="mensaje" class="mensaje">Message <span>(required)</span></label>
                <textarea name="mensaje" class="mensajes"><?php if(!$enviar && isset($mensaje)) echo $mensaje ?></textarea></s
                <?php if(!empty($error)): ?>
                    <div class="error">
                        <?php echo $error; ?>
                    </div>
                <?php elseif($enviar): ?>
                    <div class="exito">
                        <p>
                        Your message has been successfully sent
                    </p>
                </div>
        <?php endif ?>
        <input type="submit" name="submit" class="btn" value="Submit">
    </form>
</div>
</body>
</html>
```

Este archivo tiene al final de las líneas un código PHP (representado por <?php) unido a vectores como $nombre, $apellido, etc que se encuentran dentro de la etiqueta <form>, lo que se imprime con la función “echo”. Sin embargo, el código no estará completo, ni validado sin el archivo contacto.php

Cabe recalcar que a este archivo se le agregó el código HTML del index.html, para que todas las pestañas tengan el mismo encabezado.

6.8.2. “contacto.php”

Este archivo tiene la programación para la validación *per se*, se usa la función isset() porque garantiza que se reciban todos los campos requeridos y en ningún caso se envíen vacíos. “IF”, ejecuta el código si la condición es verdadera; “ELSE”, es como “si no”, es el código ejecutado si la condición es falsa; “THEN”, es como “luego” y se usa para realizar otra acción. Además, este archivo está enlazado y requiere al archivo validado_contacto.php (línea 44 de la figura de abajo). También es importante enfatizar que aquí es donde se coloca el correo que recibirá la información (línea 34) y el título del mensaje que lo va a caracterizar (línea 35).

```php
<?php
$error = '';
$enviar = '';
if (isset($_POST['submit'])) {
    $nombre = $_POST['nombre'];
    $apellido = $_POST['apellido'];
    $correo = $_POST['correo'];
    $mensaje = $_POST['mensaje'];
    if (!empty($nombre)) {
        $nombre = filter_var($nombre,FILTER_SANITIZE_STRING);
        } else {
        $error .= 'The name field is required <br>';
        }
    if (!empty($apellido)) {
        $apellido = filter_var($apellido,FILTER_SANITIZE_STRING);
        } else {
        $error .= 'The last name is required <br>';
        }
    if (!empty($correo)) {
        $correo = filter_var($correo,FILTER_SANITIZE_EMAIL);
        if (!filter_var($correo, FILTER_VALIDATE_EMAIL)) {
        $errp .= 'Please, enter a valid email <br>';
        }
        } else {
        $error .= 'The email field is required <br>';
        }
        if (!empty($mensaje)) {
            $mensaje = htmlspecialchars($mensaje);
            $mensaje = filter_var($mensaje, FILTER_SANITIZE_STRING);
            } else {
            $error .= 'The field message is empty <br>';
            }
            if (!$error) {
                $enviar_a = 'chirinos@cefobi-conicet.gov.ar';
                $asunto = 'Correo enviado desde mi página web';
                $mensaje_a_preparar = "Nombre : $nombre \n";
                $mensaje_a_preparar .= "Apellido : $apellido \n";
                $mensaje_a_preparar .= "Correo : $correo \n";
                $mensaje_a_preparar = "Mensaje : . $mensaje ";
                mail($enviar_a, $asunto , $mensaje_a_preparar);
                    $enviar = true;
                }
}
require 'validado_contacto.php';
?>
```

6.8.3. “estilo1.css”

Este archivo es solo para darle estilo al documento HTML, por ello se enlaza y se nombra por medio del atributo href en el archivo validado_contacto.php Es importante usar CSS para evitar duplicación, no colocar tanto código dentro del lenguaje de marcado y en cada página o pestaña que se cree, lo que resulta engorroso e incluso puede conllevar a errores. El uso de CSS hace el mantenimiento y actualización de la página muchos más simple.

```
1  *{
2      margin: 0;
3      padding: 0;
4  }
```

```
59 body{
60     background: rgb(255, 255, 255);
61     }
62     .contenedor{
63         padding: 50px;
64         background: #fff;
65         margin: 100px auto;
66         width: 66%;
67         border-radius: 9px;
68         border: 5px solid rgb(157, 221, 182);
69         }
70         .texto{
71             font-size: 18px;
72             margin-left: 70px;
73             }
```

De la línea 5 a la 58 corresponde al código del apartado 4 llamado encabezado.css que es el archivo que le da estilo al encabezado de todas las pestañas.

```
74         span{
75             color: rgb(236, 12, 12);
76             font-size: 12px;
77             }
78         .input{
79             width: 340px;
80             display: block;
81             margin: 10px;
82             padding: 14px;
83             border: 3px solid rgb(1, 128, 147);
84             border-radius: 8px;
```

```
        background: rgb(184, 174, 175);
        }
        .mensaje{
            margin-right: 100px;
            font-size: 18px;
            }
        .mensajes{
            width: 340px;
            display: block;
            margin: 10px;
            height: 200px;
            padding: 14px;
            border: 3px solid rgb(1, 128, 147);
            border-radius: 8px;
            background: rgb(184, 174, 175);
            }
        .btn{
            float: right;
            background: rgb(28, 4, 98);
            color: #fff;
            cursor: pointer;
            padding:10px;
            margin-top: -13px;
            border-radius: 4px;
            border: 1px;
            }
        .btn:hover{
            background: rgb(47, 3, 154);
            }
        h1{
            text-align: center;
            margin: 10px auto;
            margin-top: 50px;
            margin-bottom: -40px;
            padding: 10px;
            background: #fff;
            width: 380px;
            border-radius: 5px;
            color: rgb(46, 115, 77);
            }
```

```
        .error{
            background: rgb(227, 56, 81);
            width: 340px;
            color: #fff;
            padding: 10px;
            margin: 5px;
            border-radius: 4px;
            font-size: 16px;
            text-align: center;
            }
        .exito{
            background: rgb(51, 187, 85);
            width: 340px;
            color: #fff;
            padding: 10px;
            margin: 5px;
            border-radius: 4px;
            }
```

Las hojas de estilo .css se definen en pixeles, en este caso los colores se colocaron en formato RGB.

ANEXO 6.9: Código para la conexión de la base de datos y la realización de consultas (pruebas del sistema)

Luego de crear la base de datos dentro de la plataforma web EURECA y sus tablas, se debe conectar la BD, para ello se usa la función “mysqli_connect”, seguido del primer parámetro que es la dirección donde se encuentra el gestor de base de datos (localhost), el segundo es el nombre del usuario de la BD que por defecto es “root”, el tercero indica la clave del usuario que por defecto al instalar el XampServer es una clave vacía y el cuarto parámetro indica el nombre de la BD (línea 3 de la figura 6.9.1). Se coloca la función IF como condicional, es decir que realice la conexión y además (condición ELSE) si ocurre un fallo, imprima (función echo) conexión no exitosa.

conexion.php

```
1  <?php
2  $config = include 'config.php';
3
4  $con=  mysqli_connect( $config['db']['host'], $config['db']['user'], $config['db']['pass'], $config['db']['name']);
5  if (!$con)
6  {
7  echo "Failed to connect to MySQL: " . mysqli_connect_error();
8  }
9  else{
10 // Begin SQL query
11
12 };
```

Figura 6.9.1. Archivo conexion.php

El archivo conexión.php está conectado a config.php cuyo código se muestra en la figura 6.9.2

config.php

```
<?php

return [
  'db' => [
    'host' => '127.0.0.1',
    'user' => 'root',
    'pass' => '',
    'name' => 'Eureca',
    'options' => [
      PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    ]
  ]
];
```

Figura 6.9.2. Archivo config.php

El proceso de consulta de datos de una tabla es similar al del listado, la diferencia es que se muestra sólo aquel o aquellos que cumplen la condición

solicitada.

Para poder realizar las consultas, se crearon los archivos BER_View.php, NER_View.php y MMR_View.php que muestran las diferentes tablas (consultas). Por ejemplo, BER_View.php tiene el archivo BER_EAF.php con la tabla para factores abióticos, BER_EFB.php con la tabla para factores bióticos, BER_Genes.php con la tabla de genes, BER_Papers.php con la tabla de papers, BER_Phenotype.php con los fenotipos reportados, BER_Proteins.php con la tabla de proteínas y las tablas BER_proteinsList.php con BER_ProteinsFiltered.php para filtrar proteínas dependiendo del sistema a estudiar. Lo mismo para los sistemas NER y MMR, tal como se puede ver en la figura 6.9.3.

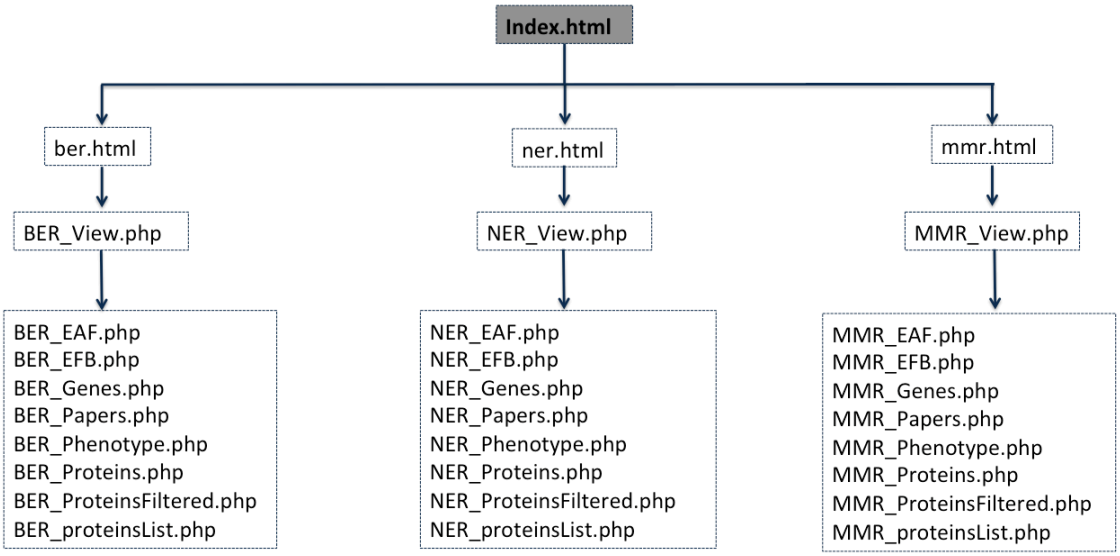


Figura 6.9.3. Arquitectura de archivos para las consultas de los sistemas de reparación indirecta

Para la consulta de especies, también se crearon carpetas con los nombres de las especies y dos archivos PHP por cada una, el archivo Papers.php para mostrar la tabla de papers y el archivo Proteins.php para mostrar la tabla de proteínas, tal como se puede ver en la figura 6.9.4.

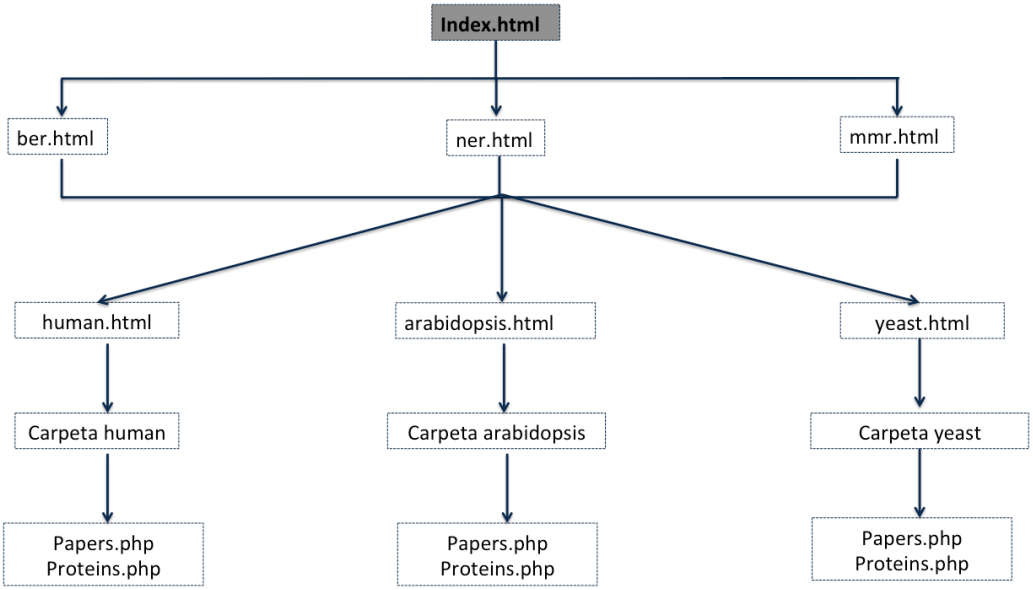
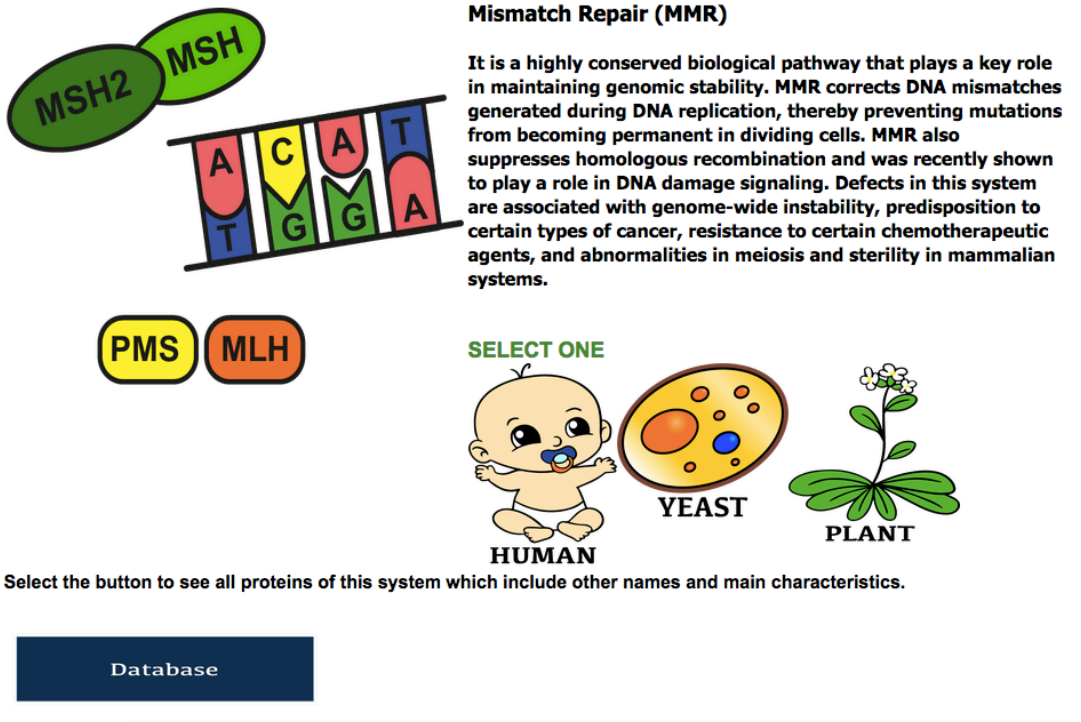


Figura 6.9.4. Arquitectura de archivos para las consultas de las especies

De esta forma, las pestañas de los diferentes sistemas de reparación indirecta quedaron con la descripción de las 7 consultas (botones azules) que están enlazados a sus respectivos archivos PHP. La figura 6.9.5 muestra la pestaña MMR.



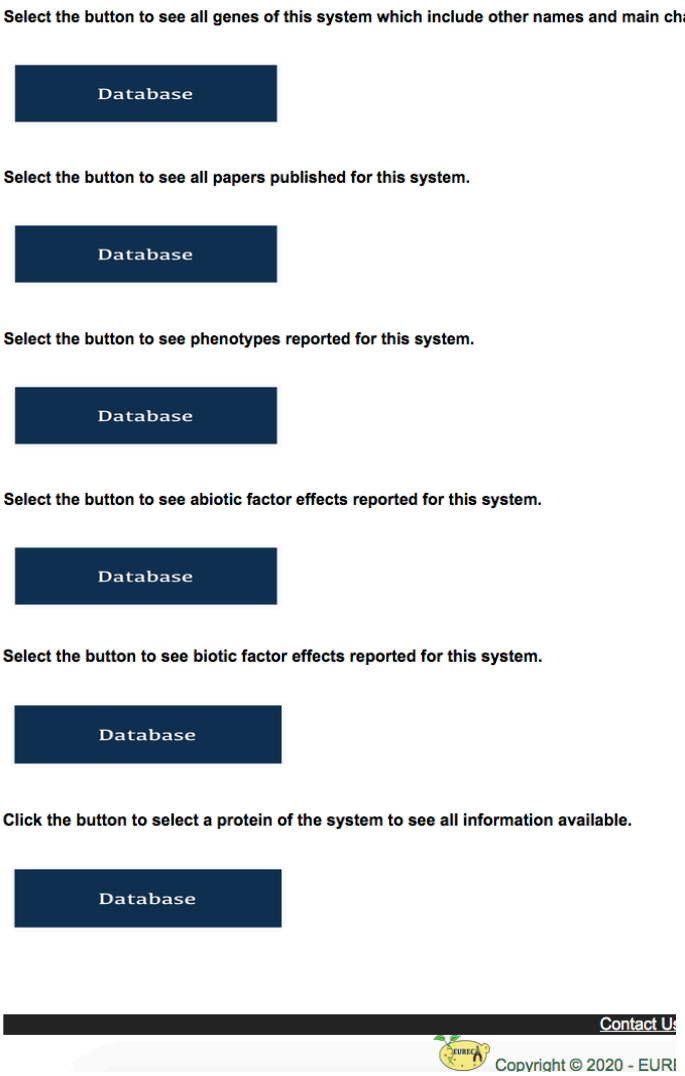


Figura 6.9.5. Pestaña mmr.html con las consultas incorporadas

Por otro lado, las pestañas de las diferentes especies quedaron con la descripción de las 2 consultas (botones azules) que están enlazados a sus respectivos archivos PHP. La figura 6.9.6 muestra la pestaña de Arabidopsis.

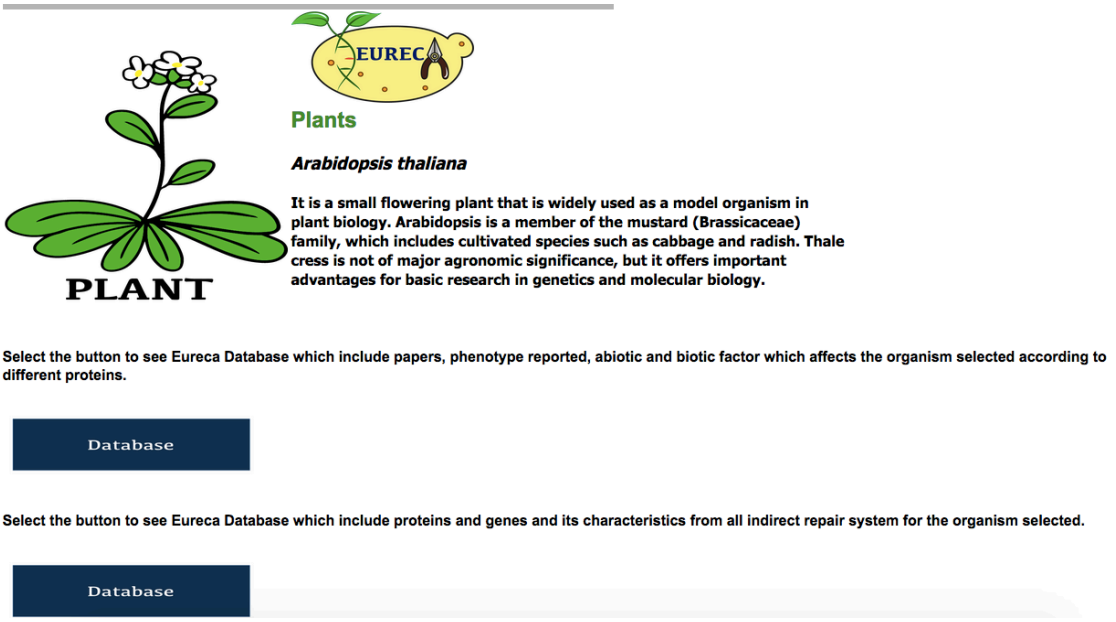


Figura 6.9.6. Pestaña arabidopsis.html con las consultas incorporadas

Para que los botones azules lleven al archivo PHP, estos se enlazaron vía href como se muestra en la figura 6.9.7 para levadura.

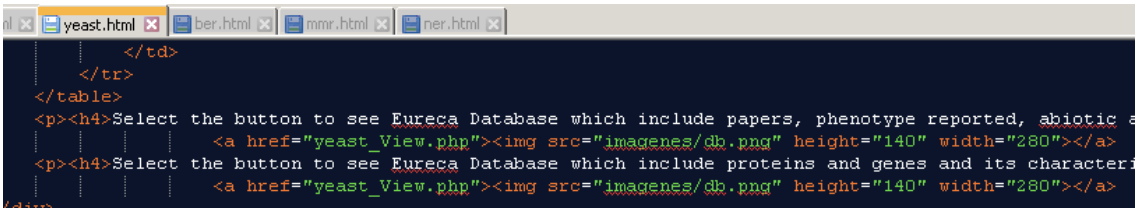


Figura 6.9.7. Pestaña yeast.html con los archivos php enlazados

o en la figura 6.9.8 para los sistemas de reparación indirectos

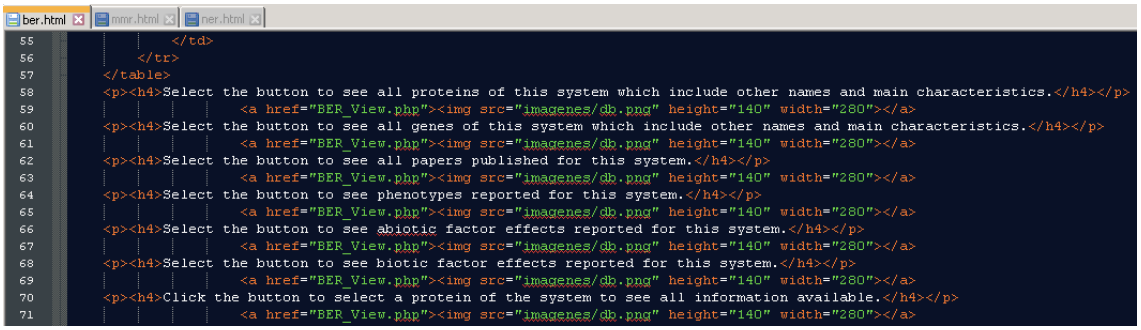


Figura 6.9.8. Pestaña ber.html con los archivos php enlazados

Los códigos PHP y SQL para las consultas fueron los siguientes:

Para los archivos principales BER_View.php, NER_View.php y MMR_View.php se muestra una estructura similar, el código para BER_View.php es el que se muestra a continuación.

```
    <?php include 'nav.php';?>
    <div class="container">


        <ul class="nav nav-tabs" id="myTab" role="tablist">
            <li class="nav-item" role="presentation">
                <button class="nav-link active" id="protein-tab" data-bs-toggle="tab" data-bs-target="#protein" type="button" role="tab" aria-
controls="protein" aria-selected="true">Proteínas</button>
            </li>
            <li class="nav-item" role="presentation">
                <button class="nav-link" id="genes-tab" data-bs-toggle="tab" data-bs-target="#genes" type="button" role="tab" aria-controls="genes"
aria-selected="false">Genes</button>
            </li>
            <li class="nav-item" role="presentation">
                <button class="nav-link" id="papers-tab" data-bs-toggle="tab" data-bs-target="#papers" type="button" role="tab" aria-controls="papers"
aria-selected="false">Papers</button>
            </li>
            <li class="nav-item" role="presentation">
                <button class="nav-link" id="phenotype-tab" data-bs-toggle="tab" data-bs-target="#phenotype" type="button" role="tab" aria-
controls="phenotype" aria-selected="false">Fenotipos</button>
            </li>
            <li class="nav-item" role="presentation">
                <button class="nav-link" id="eaf-tab" data-bs-toggle="tab" data-bs-target="#eaf" type="button" role="tab" aria-controls="eaf" aria-
selected="false">Efectos abióticos</button>
            </li>
            <li class="nav-item" role="presentation">
                <button class="nav-link" id="efb-tab" data-bs-toggle="tab" data-bs-target="#efb" type="button" role="tab" aria-controls="efb" aria-
selected="false">Efectos bióticos</button>
            </li>
            <li class="nav-item" role="presentation">
                <button class="nav-link" id="filter-tab" data-bs-toggle="tab" data-bs-target="#filter" type="button" role="tab" aria-controls="filter"
aria-selected="false">Selecciona una proteína</button>
            </li>

        </ul>

<div class="tab-content" id="myTabContent">
    <div class="tab-pane fade show active" id="protein" role="tabpanel" aria-labelledby="protein-tab">
        <div class="table-responsive">
            <table class="table table-striped table-hover" id=>
                <tr>
                    <th>N°</th>
                    <th>aanumber</th>
                    <th>fastaprot</th>
                    <th>weight</th>
                    <th>pI</th>
                    <th>instindex</th>
                    <th>alipindex</th>
                    <th>GRAVY</th>
                    <th>name1</th>
                    <th>name2</th>
                    <th>name3</th>
                    <th>IDuniprot</th>
                    <th>GI</th>
                </tr>
                <?php
                $sql = include 'BER/BER_Proteins.php';

                if(mysqli_num_rows($sql) == 0){
                    echo '<tr><td colspan="8">No hay datos.</td></tr>';
                }else{
                    $n = 1;
                    while($row = mysqli_fetch_assoc($sql)){
                        echo '
                        <tr>
                            <td>'.$n.'</td>
                            <td>'.$row['aanumber'].'</td>
                            <td>'.$row['fastaprot'].'</td>
                            <td>'.$row['weight'].'</td>
                            <td>'.$row['pI'].'</td>
                            <td>'.$row['instindex'].'</td>
                            <td>'.$row['alipindex'].'</td>
                            <td>'.$row['GRAVY'].'</td>
                            <td>'.$row['name1'].'</td>
                            <td>'.$row['name2'].'</td>
                            <td>'.$row['name3'].'</td>
                            <td>'.$row['IDuniprot'].'</td>
                            <td>'.$row['GI'].'</td>

                        </tr>
                        ';
                        $n++;
                    }
                }
                ?>
            </table>
```

```
<div class="tab-content" id="myTabContent">
    <div class="tab-pane fade show active" id="protein" role="tabpanel" aria-labelledby="protein-tab">
        <div class="table-responsive">
            <table class="table table-striped table-hover" id=>
                <tr>
                    <th>N°</th>
                    <th>aanumber</th>
                    <th>fastaprot</th>
                    <th>weight</th>
                    <th>pI</th>
                    <th>instindex</th>
                    <th>alipindex</th>
                    <th>GRAVY</th>
                    <th>name1</th>
                    <th>name2</th>
                    <th>name3</th>
                    <th>IDuniprot</th>
                    <th>GI</th>
                </tr>
                <?php
                $sql = include 'BER/BER_Proteins.php';

                if(mysqli_num_rows($sql) == 0){
                    echo '<tr><td colspan="8">No hay datos.</td></tr>';
                }else{
                    $n = 1;
                    while($row = mysqli_fetch_assoc($sql)){
                        echo '
                        <tr>
                            <td>'.$n.'</td>
                            <td>'.$row['aanumber'].'</td>
                            <td>'.$row['fastaprot'].'</td>
                            <td>'.$row['weight'].'</td>
                            <td>'.$row['pI'].'</td>
                            <td>'.$row['instindex'].'</td>
                            <td>'.$row['alipindex'].'</td>
                            <td>'.$row['GRAVY'].'</td>
                            <td>'.$row['name1'].'</td>
                            <td>'.$row['name2'].'</td>
                            <td>'.$row['name3'].'</td>
                            <td>'.$row['IDuniprot'].'</td>
                            <td>'.$row['GI'].'</td>

                        </tr>
                        ';
                        $n++;
                    }
                }
                ?>
            </table>

<div class="tab-pane fade" id="genes" role="tabpanel" aria-labelledby="genes-tab">
  ss <div class="table-responsive">
        <table class="table table-striped table-hover">
            <tr>
                <th>N°</th>
                <th>namegen</th>
                <th>fastagen</th>
                <th>function</th>
                <th>name1</th>
                <th>name2</th>
                <th>name3</th>
                <th>IDNCBI</th>
            </tr>
            <?php
            $sqlg = include 'BER/BER_Genes.php';

            if(mysqli_num_rows($sqlg) == 0){
                echo '<tr><td colspan="8">No hay datos.</td></tr>';
            }else{
                $n = 1;
                while($row = mysqli_fetch_assoc($sqlg)){
                    echo '
                    <tr>
                        <td>'.$n.'</td>
                        <td>'.$row['namegen'].'</td>
                        <td>'.$row['fastagen'].'</td>
                        <td>'.$row['function'].'</td>
                        <td>'.$row['name1'].'</td>
                        <td>'.$row['name2'].'</td>
                        <td>'.$row['name3'].'</td>
                        <td>'.$row['IDNCBI'].'</td>

                    </tr>
                    ';
                    $n++;
                }
            }
            ?>
```

```
<div class="tab-pane fade" id="papers" role="tabpanel" aria-labelledby="papers-tab">
    <div class="table-responsive">
        <table class="table table-striped table-hover">
            <tr>
                <th>N°</th>
                <th>IDpaper</th>
                <th>linkpaper</th>
                <th>author</th>
                <th>year</th>
                <th>PatGenus</th>
                <th>Patspecie</th>
                <th>Pattype</th>
                <th>stress</th>
                <th>name1</th>
            </tr>
            <?php
            $sqlp = include 'BER/BER_Papers.php';

            if(mysqli_num_rows($sqlp) == 0){
                echo '<tr><td colspan="8">No hay datos.</td></tr>';
            }else{
                $n = 1;
                while($row = mysqli_fetch_assoc($sqlp)){
                    echo '
                    <tr>
                        <td>'.$n.'</td>
                        <td>'.$row['IDpaper'].'</td>
                        <td>'.$row['linkpaper'].'</td>
                        <td>'.$row['author'].'</td>
                        <td>'.$row['year'].'</td>
                        <td>'.$row['PatGenus'].'</td>
                        <td>'.$row['Patspecie'].'</td>
                        <td>'.$row['Pattype'].'</td>
                        <td>'.$row['stress'].'</td>
                        <td>'.$row['name1'].'</td>

                    </tr>
                    ';
                    $n++;
                }
            }
            ?>
        </table>

<div class="tab-pane fade" id="phenotype" role="tabpanel" aria-labelledby="phenotype-tab">
    <div class="table-responsive">
        <table class="table table-striped table-hover">
            <tr>
                <th>N°</th>
                <th>pheno1</th>
                <th>PatGenus</th>
                <th>Patspecie</th>
                <th>Pattype</th>
                <th>stress</th>
                <th>name1</th>
            </tr>
            <?php
            $sqlph = include 'BER/BER_Phenotype.php';

            if(mysqli_num_rows($sqlph) == 0){
                echo '<tr><td colspan="8">No hay datos.</td></tr>';
            }else{
                $n = 1;
                while($row = mysqli_fetch_assoc($sqlph)){
                    echo '
                    <tr>
                        <td>'.$n.'</td>
                        <td>'.$row['pheno1'].'</td>
                        <td>'.$row['PatGenus'].'</td>
                        <td>'.$row['Patspecie'].'</td>
                        <td>'.$row['Pattype'].'</td>
                        <td>'.$row['stress'].'</td>
                        <td>'.$row['name1'].'</td>

                    </tr>
                    ';
                    $n++;
                }
            }
            ?>
        </table>
```

```
<div class="tab-pane fade" id="eaf" role="tabpanel" aria-labelledby="eaf-tab">
    <div class="table-responsive">
        <table class="table table-striped table-hover">
            <tr>
                <th>N°</th>
                <th>IDpaper</th>
                <th>linkpaper</th>
                <th>author</th>
                <th>year</th>
                <th>stress</th>
                <th>pheno1</th>
                <th>name1</th>
            </tr>
            <?php
            $sqle = include 'BER/BER_EAF.php';

            if(mysqli_num_rows($sqle) == 0){
                echo '<tr><td colspan="8">No hay datos.</td></tr>';
            }else{
                $n = 1;
                while($row = mysqli_fetch_assoc($sqle)){
                    echo '
                    <tr>
                        <td>'.$n.'</td>
                        <td>'.$row['IDpaper'].'</td>
                        <td>'.$row['linkpaper'].'</td>
                        <td>'.$row['author'].'</td>
                        <td>'.$row['year'].'</td>
                        <td>'.$row['stress'].'</td>
                        <td>'.$row['pheno1'].'</td>
                        <td>'.$row['name1'].'</td>

                    </tr>
                    ';
                    $n++;
                }
            }
            ?>
        </table>

<div class="tab-pane fade" id="efb" role="tabpanel" aria-labelledby="efb-tab">
    <div class="table-responsive">
        <table class="table table-striped table-hover">
            <tr>
                <th>N°</th>
                <th>IDpaper</th>
                <th>linkpaper</th>
                <th>author</th>
                <th>year</th>
                <th>PatGenus</th>
                <th>Patspecie</th>
                <th>Pattype</th>
                <th>pheno1</th>
                <th>name1</th>
            </tr>
            <?php
            $sqlf = include 'BER/BER_EFB.php';

            if(mysqli_num_rows($sqlf) == 0){
                echo '<tr><td colspan="8">No hay datos.</td></tr>';
            }else{
                $n = 1;
                while($row = mysqli_fetch_assoc($sqlf)){
                    echo '
                    <tr>
                        <td>'.$n.'</td>
                        <td>'.$row['IDpaper'].'</td>
                        <td>'.$row['linkpaper'].'</td>
                        <td>'.$row['author'].'</td>
                        <td>'.$row['year'].'</td>
                        <td>'.$row['PatGenus'].'</td>
                        <td>'.$row['Patspecie'].'</td>
                        <td>'.$row['Pattype'].'</td>
                        <td>'.$row['pheno1'].'</td>
                        <td>'.$row['name1'].'</td>

                    </tr>
                    ';
                    $n++;
                }
            }
            ?>
        </table>
```

```
<div class="tab-pane fade" id="filter" role="tabpanel" aria-labelledby="filter-tab">
                <div class="form-group">
    <select name="filter" class="form-control" id="filter">
                                <option value="0">Seleccionar proteina</option>
        <?php
        $proteins = include 'BER/BER_proteinsList.php';

        foreach($proteins as $pro){

        ?>
            <option value="<?= $pro['name1'] ?>" ><?= $pro['name1'] ?></option>
        <?php
        }
        ?>


                        </select>
                </div>
    <div class="table-responsive">
        <table class="table table-striped table-hover">
            <tr>
                <th>N°</th>
                <th>IDpaper</th>
                <th>linkpaper</th>
                <th>author</th>
                <th>year</th>
                <th>PatGenus</th>
                <th>Patspecie</th>
                <th>Pattype</th>
                <th>pheno1</th>
                <th>name1</th>
            </tr>
           <tbody id="table-all">
           <?php
            $sqlee = include 'BER/BER_ProteinsFiltered.php';
            if(mysqli_num_rows($sqlee) == 0){
                echo '<tr><td colspan="8">No hay datos.</td></tr>';
            }else{
                $n = 1;
                while($row = mysqli_fetch_assoc($sqlee)){
                    echo '
                    <tr id = '.$row['name1'].'>
                        <td>'.$n.'</td>
                        <td>'.$row['IDpaper'].'</td>
                        <td>'.$row['linkpaper'].'</td>
                        <td>'.$row['author'].'</td>
                        <td>'.$row['year'].'</td>
                        <td>'.$row['PatGenus'].'</td>
                        <td>'.$row['Patspecie'].'</td>
                        <td>'.$row['Pattype'].'</td>
                        <td>'.$row['pheno1'].'</td>
                        <td>'.$row['name1'].'</td>

                    </tr>
                    ';
                    $n++;
                }
```

```
<script>
    $(document).ready(function () {
        var body =  $('#table-all').find('tr').toArray();
        $('select').on('change', function() {
            var protein = $(this).val();

            var result = null;
            result = body.filter(function (tr) {
                return $(tr).attr('id') === protein;
            })

            $('#table-all').html('');
            $('#table-all').append(result);
        })
    })
</script>
```

Código similar para las pestañas NER_View.php y MMR_View.php

Para las consultas se usó el siguiente código:

1. Para la pestaña BER/NER/MMR que corresponden a los 3 tipos de sistemas indirectos de reparación de ADN. Aquí al darle click aparecen los resultados de la consulta realizada.

a) Muestra todas las proteínas (nombre, características, otros nombres) del sistema BER/NER/MMR (según la pestaña seleccionada por el usuario)

```
<?php
include("conexion.php");


    $query = "SELECT pc.aanumber, pc.fastaprot, pc.weight, pc.pI , pc.instindex ,
pc.alipindex , pc.GRAVY ,
    p.name1 , p.name2 , p.name3, p2.IDuniprot , p2.GI
    FROM ProteinChar pc
    inner join Protname p on pc.nameprot = p.IDname
    inner join Proteins p2 on pc.id_prot = p2.IDuniprot
    inner join IndSyst is2 on p2.IDprot = is2.idprot
    WHERE is2.`type` ='BER'
    GROUP BY pc.aanumber, pc.fastaprot, pc.weight, pc.pI , pc.instindex , pc.alipindex ,
pc.GRAVY ,
    p.name1 , p.name2 , p.name3, p2.IDuniprot , p2.GI
    ";

    $sql = mysqli_query($con, $query);

    if(false===$sql ){
        return null;
    }else{
        return $sql;
    }
```

b) Muestra todos los genes (nombre, características, otros nombres) del sistema BER/NER/MMR (según la pestaña seleccionada por el usuario)

```
<?php
include("conexion.php");


    $query = "SELECT gc.namegen , gc.fastagen , gc.`function`,
    g.name1 , g.name2 , g.name3 , g2.IDNCBI
    from GenChar gc
    inner join Genname g on gc.ID_char = g.idcchar
    inner join Gene g2 on g.idcchar = g2.id_char
    inner join IndSyst is2 on g2.id_prot = is2.idprot
    WHERE is2.`type` ='BER'
    GROUP BY gc.namegen , gc.fastagen , gc.`function`,
    g.name1 , g.name2 , g.name3 , g2.IDNCBI
    ";

    $sql = mysqli_query($con, $query);

    if(false===$sql ){
        return null;
    }else{
        return $sql;
    }
```

c) Muestra todos los artículos científicos (autor y año), así como sus efectos por estrés biótico y abiótico para el sistema BER/NER/MMR especificando las proteínas.

```
<?php
include("conexion.php");


    $query = "SELECT p.IDpaper , p.linkpaper , p.author , p.`year` ,
    bf.PatGenus , bf.Patspecie , bf.Pattype,
    af.stress ,
    p2.name1
    FROM Papers p
    left join BioticF bf ON p.biot = bf.IDpat
    left join AbioticF af ON p.abi = af.IDstress
    inner join IndSyst is2 on p.IDpaper = is2.Idpap
    inner join Proteins p3 on is2.idprot = p3.IDprot
    inner join Protname p2 on p3.id_name = p2.IDname
    WHERE is2.`type` ='BER'
    GROUP BY p.IDpaper , p.linkpaper , p.author , p.`year` ,
    bf.PatGenus , bf.Patspecie , bf.Pattype,
    af.stress ,
    p2.name1
    ";

    $sql = mysqli_query($con, $query);

    if(false===$sql ){
        return null;
    }else{
        return $sql;
    }
```

d) Muestra todos los fenotipos reportados para los estreses bióticos y abióticos probados especificando las proteínas.

```
<?php
include("conexion.php");


    $query = "SELECT p.pheno1 ,
    bf.PatGenus , bf.Patspecie , bf.Pattype ,
    af.stress ,
    p2.name1
    FROM Phenotype p
    left join BioticF bf on p.IDphenot = bf.id_phe
    left join AbioticF af on p.IDphenot = af.id_phe
    left join IndSyst is2 on (af.IDstress = is2.id_af or bf.IDpat = is2.id_bf )
    left join Proteins p3 on is2.idprot = p3.IDprot
    left join Protname p2 on p3.id_name = p2.IDname
    WHERE is2.`type` ='BER'
    GROUP BY p.pheno1 ,
    bf.PatGenus , bf.Patspecie , bf.Pattype ,
    af.stress ,
    p2.name1
    ";

    $sql = mysqli_query($con, $query);

    if(false===$sql ){
        return null;
    }else{
        return $sql;
    }
```

e) Muestra todos los efectos abióticos (paper, fenotipo reportado, proteínas, especies) para el sistema BER/NER/MMR

```php
<?php
include("conexion.php");


    $query = "SELECT p.pheno1 ,
    bf.PatGenus , bf.Patspecie , bf.Pattype ,
    af.stress ,
    p2.name1
    FROM Phenotype p
    left join BioticF bf on p.IDphenot = bf.id_phe
    left join AbioticF af on p.IDphenot = af.id_phe
    left join IndSyst is2 on (af.IDstress = is2.id_af or bf.IDpat = is2.id_bf )
    left join Proteins p3 on is2.idprot = p3.IDprot
    left join Protname p2 on p3.id_name = p2.IDname
    WHERE is2.`type` ='BER'
    GROUP BY p.pheno1 ,
    bf.PatGenus , bf.Patspecie , bf.Pattype ,
    af.stress ,
    p2.name1
    ";

    $sql = mysqli_query($con, $query);

    if(false===$sql ){
        return null;
    }else{
        return $sql;
    }
```

f) Muestra todos los efectos bióticos (paper, fenotipo reportado, proteínas, especies) para el sistema BER/NER/MMR

```php
<?php
include("conexion.php");


    $query = "SELECT p.IDpaper , p.linkpaper , p.author , p.`year` ,
    bf.PatGenus , bf.Patspecie , bf.Pattype,
    p4.pheno1 ,
    p2.name1
    FROM Papers p
    left join BioticF bf ON p.biot = bf.IDpat
    left join Phenotype p4 on bf.id_phe = p4.IDphenot
    inner join IndSyst is2 on p.IDpaper = is2.Idpap
    inner join Proteins p3 on is2.idprot = p3.IDprot
    inner join Protname p2 on p3.id_name = p2.IDname
    WHERE is2.`type` ='BER'
    GROUP BY p.IDpaper , p.linkpaper , p.author , p.`year` ,
    bf.PatGenus , bf.Patspecie , bf.Pattype,
    p4.pheno1,
    p2.name1
    ";

    $sql = mysqli_query($con, $query);

    if(false===$sql ){
        return null;
    }else{
        return $sql;
    }
```

Por otro lado, también aparece en todas las pestañas BER/NER/MMR “selecciona una proteína” (en un botón despegable aparecen todas las proteínas de la base de datos) y al elegirla, aparecen los artículos científicos, el tipo de estrés (biótico o abiótico) y el fenotipo reportado, para lo cual se usó el siguiente código:

BER_proteinsList.php

```php
<?php
include("conexion.php");


    $query = "SELECT p.name1 from Protname p
    inner join Proteins p3 on p.IDname = p3.id_name
    inner join IndSyst is2 on p3.IDprot = is2.idprot
    WHERE is2.`type` ='BER'
    GROUP BY p.name1
    ";

    $sql = mysqli_query($con, $query);

    if(false===$sql ){
        return null;
    }else{
        return $sql;
    }
```

BER_ProteinsFiltered.php

```php
<?php
include("conexion.php");


    $query = "SELECT p.IDpaper , p.linkpaper , p.author , p.`year` ,
    af.stress ,
    bf.PatGenus , bf.Patspecie , bf.Pattype,
    p4.pheno1 ,
    p2.name1
    FROM Papers p
    left join AbioticF af ON p.abi = af.IDstress
    left join BioticF bf ON p.biot = bf.IDpat
    left join Phenotype p4 on bf.id_phe = p4.IDphenot
    inner join IndSyst is2 on p.IDpaper = is2.Idpap
    inner join Proteins p3 on is2.idprot = p3.IDprot
    inner join Protname p2 on p3.id_name = p2.IDname
    WHERE is2.`type` ='BER'
    GROUP BY p.IDpaper , p.linkpaper , p.author , p.`year` ,
    af.stress ,
    bf.PatGenus , bf.Patspecie , bf.Pattype,
    p4.pheno1 ,
    p2.name1
    ";


    $sql = mysqli_query($con, $query);
    if(false===$sql ){
        return "x";
    }else{
        return $sql;
    }
```

2. Para los archivos principales arabidopsis_View.php, human_View.php y yeast_View.php se muestra una estructura similar, el código para arabidopsis_View.php es el que se muestra a continuación.

```
                        <ul class="nav nav-tabs" id="myTab" role="tablist">
                <li class="nav-item" role="presentation">
                        <button class="nav-link active" id="protein-tab" data-bs-toggle="tab" data-bs-target="#protein" type="button"
role="tab" aria-controls="protein" aria-selected="true">Proteínas</button>
                </li>
                <li class="nav-item" role="presentation">
                        <button class="nav-link" id="papers-tab" data-bs-toggle="tab" data-bs-target="#papers" type="button" role="tab"
aria-controls="papers" aria-selected="false">Papers</button>
                </li>
```

```
                    </ul>

                    <div class="tab-content" id="myTabContent">
            <div class="tab-pane fade show active" id="protein" role="tabpanel" aria-labelledby="protein-tab">
                    <div class="table-responsive">
                    <table class="table table-striped table-hover" id=>
                    <tr>
                    <th>N°</th>
                    <th>aanumber</th>
                    <th>fastaprot</th>
                    <th>weight</th>
                    <th>pI</th>
                    <th>instindex</th>
                    <th>alipindex</th>
                    <th>GRAVY</th>
                    <th>name1</th>
                    <th>name2</th>
                    <th>name3</th>
                    <th>IDuniprot</th>
                    <th>GI</th>
                    </tr>
                    <?php
                    $sql = include 'arabidopsis/Proteins.php';

                if(mysqli_num_rows($sql) == 0){
                echo '<tr><td colspan="8">No hay datos.</td></tr>';
                }else{
                $n = 1;
                while($row = mysqli_fetch_assoc($sql)){
                echo '
                <tr>
                <td>'.$n.'</td>
                <td>'.$row['aanumber'].'</td>
                <td>'.$row['fastaprot'].'</td>
                <td>'.$row['weight'].'</td>
                <td>'.$row['pI'].'</td>
                <td>'.$row['instindex'].'</td>
                <td>'.$row['alipindex'].'</td>
                <td>'.$row['GRAVY'].'</td>
                <td>'.$row['name1'].'</td>
                <td>'.$row['name2'].'</td>
                <td>'.$row['name3'].'</td>
                <td>'.$row['IDuniprot'].'</td>
                <td>'.$row['GI'].'</td>

                </tr>
                ';
                $n++;
                }
                }
                ?>
                </table>
                </div>


<div class="tab-pane fade" id="papers" role="tabpanel" aria-labelledby="papers-tab">
                    <div class="table-responsive">
                    <table class="table table-striped table-hover">
                    <tr>
                    <th>N°</th>
                    <th>IDpaper</th>
                    <th>linkpaper</th>
                    <th>author</th>
                    <th>year</th>
                    <th>stress</th>
                    <th>PatGenus</th>
                    <th>Patspecie</th>
                    <th>Pattype</th>
                    </tr>
                    <?php
                    $sqlp = include 'arabidopsis/Papers.php';

                    if(mysqli_num_rows($sqlp) == 0){
                    echo '<tr><td colspan="8">No hay datos.</td></tr>';
                    }else{
                    $n = 1;
                    while($row = mysqli_fetch_assoc($sqlp)){
                    echo '
                    <tr>
                    <td>'.$n.'</td>
                    <td>'.$row['IDpaper'].'</td>
                    <td>'.$row['linkpaper'].'</td>
                    <td>'.$row['author'].'</td>
                    <td>'.$row['year'].'</td>
                    <td>'.$row['stress'].'</td>
                    <td>'.$row['PatGenus'].'</td>
                    <td>'.$row['Patspecie'].'</td>
                    <td>'.$row['Pattype'].'</td>

                    </tr>
                    ';
                    $n++;
                    }
                    }
                    ?>
                    </table>
                    </div>
```

```
                        <script src="https://code.jquery.com/jquery-3.3.1.js"></script>
                        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

                        </body>

                        </body>
                        </html>

                        <script>
                        $(document).ready(function () {
                        var body =  $('#table-all').find('tr').toArray();
                        $('select').on('change', function() {
                        var protein = $(this).val();

                        var result = null;
                        result = body.filter(function (tr) {
            return $(tr).attr('id') === protein;
                        })

                        $('#table-all').html('');
                        $('#table-all').append(result);
                        })
                        })
                        </script>
```

a) Muestra todas las proteínas y genes reportados para Arabidopsis/humano/levadura

```
<?php
include("conexion.php");


    $query = "SELECT  pc.aanumber , pc.fastaprot , pc.weight , pc.pI , pc.instindex ,
pc.alipindex , pc.GRAVY ,
    p.name1 , p.name2 , p.name3 ,
    p2.IDuniprot , p2.GI
    FROM ProteinChar pc
    left join Protname p on pc.nameprot = p.IDname
    left join Proteins p2 on pc.nameprot = p2.id_name
    left join Species s on p2.IDprot  = s.id_pro
    where s.namesp = 'arabidopsis'
    GROUP BY pc.aanumber , pc.fastaprot , pc.weight , pc.pI , pc.instindex ,
pc.alipindex , pc.GRAVY ,
    p.name1 , p.name2 , p.name3 ,
    p2.IDuniprot , p2.GI
    ";

    $sql = mysqli_query($con, $query);

    if(false===$sql ){
        return null;
    }else{
        return $sql;
    }
```

b) Muestra todos los artículos científicos, fenotipo, factores bióticos o abióticos reportados para Arabidopsis/humano/levadura

```
<?php
include("conexion.php");


    $query = "SELECT p.IDpaper , p.linkpaper , p.author , p.`year` ,
    af.stress ,
    bf.PatGenus , bf.Patspecie , bf.Pattype
    FROM Papers p
    left join AbioticF af ON p.abi = af.IDstress
    left join BioticF bf ON p.biot = bf.IDpat
    left join Species s on p.idsp  = s.IDsp
    where s.namesp = 'arabidopsis'
    GROUP BY p.IDpaper , p.linkpaper , p.author , p.`year` ,
    af.stress ,
    bf.PatGenus , bf.Patspecie , bf.Pattype
    ";

    $sql = mysqli_query($con, $query);

    if(false===$sql ){
        return null;
    }else{
        return $sql;
    }
```

Código similar para las pestañas human_View.php y yeast_View.php. Pueden ver el funcionamiento de la plataforma web EURECA con datos ficticios, que luego serán reemplazados con los reales en https://www.youtube.com/watch?v=v-qxPRO1ssQ