TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Zacatepec

“API PARA EL DESPLIEGUE DINÁMICO DE LOS ELEMENTOS GENÉTICOS EN EL DNA, EN LA PÁGINA DE GENES DE LA BASE DE DATOS REGULONDB”

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

PRESENTA:

LUIS OLARTE GERVACIO

ZACATEPEC, MORELOS **MARZO , 2018**

Copyright © 2017 por Luis Olarte Gervacio. Todos los derechos reservados.

DEDICATORIA

A toda mi familia por sus buenos deseos y consejos que me dieron desde el inicio de la ingeniería especialmente a mis padres que me dieron y se preocuparon por que nada me faltara, gracias por toda la educación, valores, apoyo y conocimiento que siempre me brindaron.

AGRADECIMIENTOS

Gracias al Centro de Ciencias Genómicas por brindar la oportunidad y confianza de realizar uno de sus proyectos y por todas las atenciones que me fueron presentadas, porque siempre fue un sueño pertenecer a la Universidad Nacional Autónoma de México “Ser orgullosamente UNAM”.

Infinito agradecimiento a la L.I. Heladia Salgado Osorio y a M.T.I. Luis José Muñiz Rascado quienes desde el primer momento me ayudaron a culminar esta última fase de mi carrera.

Agradecimiento especial para M.T.I. Kevin Alquicira Hernández y M.T.I. Shirley Alquicira Hernández por el apoyo y guía durante el desarrollo del proyecto.

Al lector, como agradecimiento por tomarse el tiempo de leer estas palabras.

Tabla de contenidos

Lista de tablas

Lista de figuras

CAPÍTULO 1: INTRODUCCIÓN

La comunicación humana es el área dedicada a entender cómo se comunican los seres humanos. La razón de ser de la comunicación es transmitir información, sea para manifestar sentimientos, influir en los demás o para realizar acciones específicas (Wikipedia).

La comunicación refuerza las relaciones sociales, enriquece a sus participantes y es el principal agente del desarrollo cultural. Suelen usarse diversos instrumentos para lograr la comunicación, como, por ejemplo, el lenguaje verbal, las imágenes, gestos, movimientos, miradas, etc. Pero podemos decir que, la humanidad encontró en las imágenes y después en las palabras, dos poderosas herramientas que han hecho posible la transmisión del conocimiento de generación en generación.

Conforme se van haciendo nuevos descubrimientos y a la vez nuevas interrogantes, la humanidad va encontrado nuevas áreas de estudio cada vez más especializadas y complejas. Ejemplo de ello son la genética, la biología molecular, la biotecnología, la ingeniería genética, la bioquímica, etc.

Por ejemplo, la genética, es el área de estudio de la biología que busca comprender y explicar cómo se transmite la herencia biológica de generación en generación. Los procesos implicados y las moléculas que intervienen en esta actividad son formas microscópicas que interactúan entre sí, invisibles al ojo humano, lo que complica aún más el comunicar el conocimiento o hallazgos a otros. Por lo que la representación gráfica ha resultado ser en esta área uno de las formas de comunicación más significativas.

De hecho, la observación de esos procesos y moléculas mediante el uso de herramientas de visualización especializadas, logran generar inscripciones o representaciones de esa realidad invisible. Los microscopios electrónicos, por ejemplo, regalan las formas de los cromosomas, la cristalografía de rayos X reveló la estructura del ADN 3D, los datos de microarrays expresan los genes en coloridos heat-maps y las tecnologías de secuenciación convierten los genomas de los organismos en un sinfín de letras (book: Han Yun. Communicating Genetics: Visualizations and Representations)[1]

Es responsabilidad del investigador, poder transmitir el conocimiento primero a sus iguales, pero también al resto de la humanidad, de formas compresibles sin el uso del lenguaje especializado para lograr una comprensión de la información transmitida.

El diseño de información se ocupa de la representación gráfica de datos, conceptos, procesos y relaciones complejas y multimodales, de forma que se aliente a la recuperación de información y la toma de decisiones. El término está estrechamente relacionado con, y a menudo se usa de manera intercambiable con, visualización de la información, o simplemente estudios de visualización. El marco ha sido utilizado para examinar diversos artefactos visuales, incluyendo ilustraciones científicas, diagramas técnicos, instrucciones visuales, infografías y mapas (Wikipedia)[2].

Dada la complejidad del conocimiento en genética, se hace uso de técnicas o herramientas como el diseño de la información para poder representar los diversos procesos, moléculas y conceptos que se quieren explicar.

Según Han Yun, en su libro Communicating Genetics: Visualizations and Representations, menciona que la genética clásica empleaba iconos minimalistas para representar el conocimiento, pero esta estrategia estaba perdiendo su capacidad de crear ilustraciones adecuadas, debido al aumento del conocimiento, a la complejidad de los procesos y la abstracción visual. Aunque menciona, que las formas simples, informales y poco detalladas pueden ser una buena forma de

comunicar, pero hay que evitar el minimalismo que quita atractivo visual, esencial para transmitir el conocimiento.

Esta tesis se desarrolla bajo el marco de la representación gráfica de la regulación genética en la bacteria de Escherichia coli K12. La iconografía implementada en esta tesis, ha sido obtenida por el grupo de expertos dedicados a la revisión de artículos científicos, que han compilado la iconografía utilizada en el área, o en caso de no existir, se han dado a la tarea de crear el diseño del componente a representar.

1.1 Antecedentes

El Programa de Genómica Computacional (PCG), ubicado en el Centro de Ciencias Genómicas (CCG) de la Universidad Nacional Autónoma de México

(UNAM), está enfocado principalmente a la investigación científica de la bacteria "*Escherichia coli* K-12[1]".

Uno de los proyectos del PGC es RegulonDB[3], la cual es una base de datos enfocada a la recopilación de información sobre la regulación que se dá en el proceso de la transcripción génica en la bacteria *Escherichia coli K12* (*E. coli* K-12), y que a través del tiempo ha sido enriquecida al conocerse más detalles de éste complejo proceso, lo que ha implicado el rediseño de su modelo de datos para integrar la nueva información[3-12].

Escherichia coli es la bacteria modelo usada en una gran cantidad de experimentos genéticos, debido a que es un organismo cuyo crecimiento es rápido y su cultivo es sencillo. Es de las bacterias más estudiadas por el ser humano, debido al impacto en la salud. Miles de artículos se han publicado en relación a algún mecanismo de esta bacteria y la respuesta o reacción de aplicar antibióticos, algún otra sustancia o molécula para estudiar su comportamiento.

[1] Es una bacteria que se encuentra en el intestino grueso de los organismos.

Inicialmente RegulonDB, se enfocó a la búsqueda de artículos científicos que describieran algún componente de la regulación genética, pero con el tiempo se ha ido integrando más información del contexto de este mecanismo. RegulonDB cuenta con un grupo de biólogos a los que se denominan curadores, que son los encargados de extraer ese conocimiento de los artículos científicos para resguardarlo en la base de datos. Éste proceso inicia realizando búsquedas en PubMed [3], que es un repositorio que indexa los resúmenes o abstracts de los artículos publicados en revistas científicas, posteriormente los curadores descargan el artículo completo y extraen la información que se necesita.

Actualmente se puede consultar la base de datos RegulonDB a través de su sitio web regulondb.ccg.unam.mx (ver Figura 1), el cuál es un portal que se ha ido enriqueciendo con el tiempo, donde se muestra información textual y gráfica del mecanismo de regulación genética. Cabe destacar que es la base de datos más completa sobre este mecanismo por lo que es consultada por gente de todo el mundo.

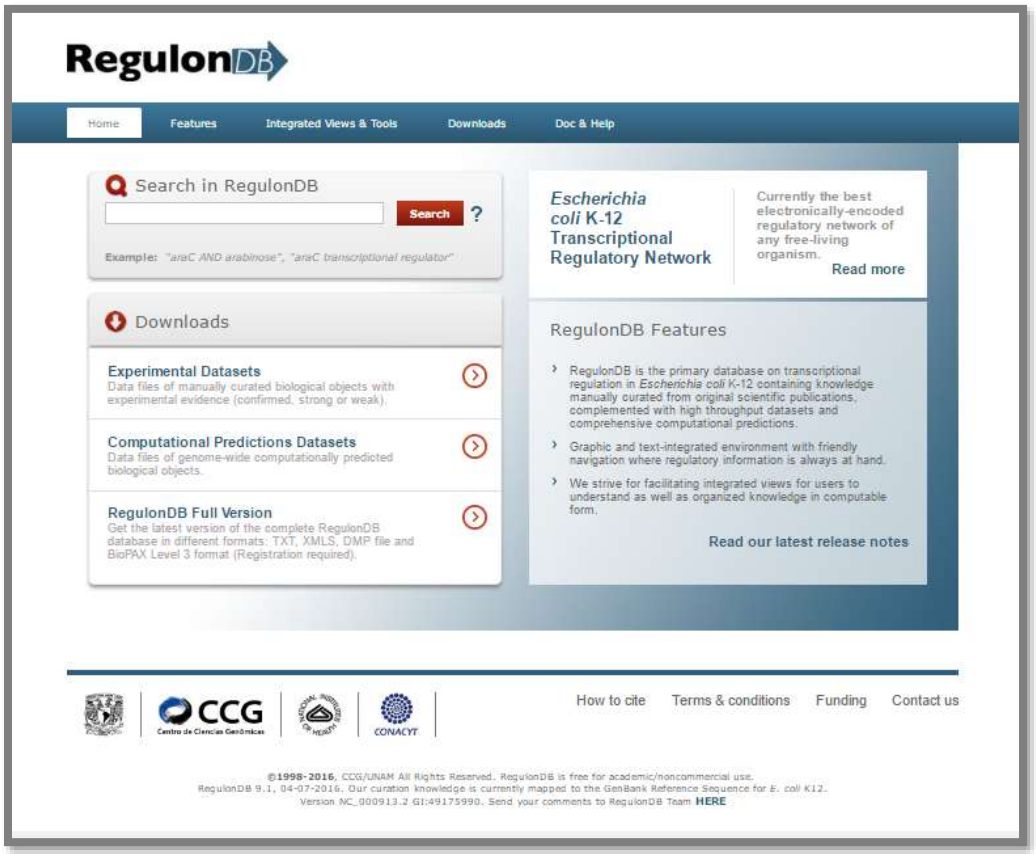


Figura 1:

Página principal de RegulonDB (http://regulondb.ccg.unam.mx)

1.2 Planteamiento del problema

RegulonDB[3] es una base de datos sobre la regulación transcripcional de la bacteria *E. coli*, ofreciendo a los investigadores a nivel internacional un portal actualizado sobre este tema. La base de datos es consultada por el web (regulondb.ccg.unam.mx), y cuenta con una interfaz con diferentes tipos de

búsquedas, como son genes, operones, condiciones de crecimiento, reguladores transcripcionales, etc.

Varias de las páginas de resultados de una búsqueda muestran despliegues gráficos estáticos de los elementos genéticos en el DNA, por ejemplo, la página de gene muestra una imagen del contexto genómico del gene, cuya imagen ha sido previamente calculada usando un programa llamado DrawingTracesTool implementado en java, y las imágenes son insertadas en el resultado de una búsqueda (ver figura 2).

De hecho, se generan en cada libración o release de datos, todas las imágenes que son usadas en la interfaz web de RegulonDB, por lo que la aplicación web requiere más espacio en disco para almacenar todas las imágenes.

El proyecto consiste en implementar que éstos gráficos del contexto genómico de un gene puedan ser generados de manera dinámica, es decir creados cuando se solicite una búsqueda.

Por lo tanto, es necesario **el desarrollo de un API para el despliegue dinámico de los elementos genéticos para la página web de resultados de la búsqueda de genes en RegulonDB y que pueda ser interpretada por los**

navegadores web, para reemplazar las imágenes almacenadas actualmente dentro de la aplicación web.

Figura 2: *Pagina de Gene en RegulonDB.*

Además se requiere que el API sea desarrollado siguiendo en lo posible, la metodología de MoProSoft (NMX-I-059/02-NYCE[2]), esto con el fin de tener buena

[2] Establece los requisitos de los procesos a implantar en la organización a través del Modelo de Procesos de Software (MoProSoft.)

calidad de software, dejar la documentación necesaria y la capacidad de poder darle matentimiento.

1.3 Objetivo General

Reducir el tiempo del proceso de liberación de RegulonDB eliminando la creación de imágenes, lo que permitirá hacer más ligera la aplicación web eliminando las más de 8 mil imágenes que se integran actualmente. Además de integrar funcionalidades a las imágenes.

1.4 Objetivos Específicos

1. Contar con un API que sea la librería gráfica web de referencia de RegulonDB sobre la representación gráfica de la regulación genética.
2. Desplegar el contexto genómico de un gene, en la página de resultados de la búsqueda por gene, usando el API; eliminando así todas las imágenes estáticas integradas en la aplicación.
3. Contar con una herramienta gráfica con tecnología moderna, al utilizar html5 y JavaScript.
4. Disminuir el tiempo y la cantidad de errores en cada proceso de liberación de datos al eliminar la tarea de creación de imágenes.

5. Crear imágenes de calidad, con buena resolución que los usuarios puedan descargar, además de permitir varios niveles de zoom.

1.5 Justificación

Como hemos mencionado, RegulonDB es una base de datos consultada a nivel internacional, ya que almacena información de la regulación genética del organismo más usado en genética, *Escherichia coli* K12. Existe un proceso de curación que inicia con la búsqueda, selección y revisión de artículos científicos enfocados a este mecanismo.

RegulonDB genera entre tres o cuatro liberaciones de datos al año, y en ese proceso de revisión de datos, una vez validados, se procede a la generación de imágenes del contexto genómico de cada gene ejecutando la herramienta DrawingTracesTool desarrollado en Java. Una vez generadas, se valida que se hayan creado de manera correcta, y se integran un poco más de ocho mil imágenes a la aplicación para posteriormente seguir con otros pasos del proceso de liberación. Integrar imágenes no es el mecanismo adecuado, ya que la aplicación ha crecido, porque no son las únicas imágenes que se integran.

Por otro lado, el sitio web de RegulonDB se ha ido modernizando, no así sus figuras o representaciones gráficas, ya que aún manejamos la integración de imágenes estáticas en formato jpg o png, cuya única funcionalidad es la

integración de un zoom, donde al aumentarlo la imagen pierde calidad, y los elementos biológicos desplegados pierden nitidez. Así que nos vimos en la necesidad de generar imágenes que permitieran ver más de cerca la región reguladora de un gene, por eso por cada gene se generan 2 archivos gráficos.

Consideramos esencial e impostergable la creación de un API o librería gráfica para RegulonDB, que sea capaz de dibujar los elementos genéticos que son anotados en RegulonDB, y sea a través de ella se grafique el contexto genómico de los genes en el momento en que se consultan.

Otra ventaja de generar una imagen a partir del API es que, al aumentar el tamaño de esta, no pierde calidad así que se podrá permitir zooms de varios niveles.

1.6 Alcances y Limitaciones

1.6.1 Alcances

1. Crear una nueva versión de la herramienta DrawingTracesTool v2.0 (DTTv2.0) que permita generar archivos html5 haciendo uso de las funciones de la librería gráfica.

2. Desarrollar un API (DrawGraphics.js) capaz de dibujar los elementos de la regulación genética.
3. Desarrollar el proyecto siguiendo la metodología de Ingeniería de Software, con lo que se logrará que la nueva herramienta sea de calidad y susceptible a futuras actualizaciones.

1.6.2 Limitaciones

1. Los usuarios finales deben permitir la ejecución de código JavaScript en sus navegadores web, de lo contrario no se podrá generar la imagen.
2. Los navegadores web deben estar actualizados a su versión más reciente para tener un buen rendimiento.

1.7 Estructura de la tesis

El presente documento de tesis está organizado en **NUMERO DE CAPÍTULOS** que a continuación se describen.

Capítulo 2. Presenta un panorama general e introductorio a la genética, así como las herramientas y tecnologías utilizadas en este proyecto.

Capítulo 3. Describe el análisis hecho al problema.

Capítulo 4. Puntualiza el diseño de solución del proyecto.

Capítulo 5. Detalla el desarrollo realizado.

Capítulo 6: Pormenorizar las pruebas realizadas a la aplicación desarrollada.

Capítulo 7: Presenta las conclusiones obtenidas de este trabajo de tesis, así como posibles líneas de investigación futuras.

CAPÍTULO 2

MARCO TEÓRICO

En este capítulo se describen algunos conceptos básicos para entender las tecnologías que se van a usar, y posteriormente se describen las tecnologías que fueron utilizadas para la implementación del proyecto. Primeramente, entraremos con el marco computacional y seguido de ello el marco biológico.

2.1 Marco Computacional

2.1.1 Sistema

Un sistema se define como un conjunto de mecanismos y herramientas que permiten la creación e interconexión de componentes de software, junto con una colección de servicios para facilitar las labores de los componentes que residen y se ejecutan en él [13].

2.1.2 Lenguaje de Programación

Un lenguaje de programación, según Wikipedia, es un lenguaje formal diseñado para realizar procesos que pueden ser ejecutados por las computadoras. Está formado por un conjunto de símbolos y reglas sintácticas y

semánticas que definen su estructura y el significado de sus elementos y expresiones [14].

2.1.3 Desarrollo WEB

2.1.3.1 HTML5

HTML5 no es una nueva versión del antiguo lenguaje de etiquetas, ni siquiera una mejora de esta ya antigua tecnología, sino un nuevo concepto para la construcción de sitios web y aplicaciones en una era que combina dispositivos móviles, computación en la nube y trabajos en red.

HTML5 es, de hecho, una mejora de esta combinación, el pegamento que une todo. HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas. A partir de ahora, HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y JavaScript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales[15].

2.1.3.2 CSS

Las hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Stylesheets) es un lenguaje de diseño para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy utilizado para establecer el diseño visual de las páginas web, e interfaces de usuario escritas en HTML o XHTML [16].

Este lenguaje es, de hecho, un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML. Al comienzo, atributos dentro de las etiquetas HTML proveían estilos esenciales para cada elemento, pero a medida que el lenguaje evolucionó, la escritura de códigos se volvió más compleja y HTML por sí mismo no pudo satisfacer las demandas de diseñadores. En consecuencia, CSS pronto fue adoptado como la forma de separar la estructura de la presentación. Desde entonces, CSS ha crecido y ganado importancia, pero siempre desarrollado en paralelo, enfocado en las necesidades de los diseñadores y apartado del proceso de evolución de HTML.

La versión 3 de CSS sigue el mismo camino, pero esta vez con un mayor compromiso. La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño. Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta es la razón por la que cada vez que

mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas [15].

2.1.3.2 JavaScript

JavaScript es un lenguaje interpretado usado para múltiples propósitos, pero solo considerado como un complemento hasta ahora. Una de las innovaciones que ayudó a cambiar el modo en que vemos JavaScript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento de código. La clave de los motores más exitosos fue transformar el código JavaScript en código máquina para lograr velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio. Esta mejorada capacidad permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje JavaScript como la mejor opción para la web.

Para aprovechar esta prometedora plataforma de trabajo ofrecida por los nuevos navegadores, JavaScript fue expandido en relación con portabilidad e integración. A la vez, interfaces de programación de aplicaciones (APIs) fueron incorporadas por defecto en cada navegador para asistir al lenguaje en funciones elementales. Estas nuevas APIs (como Web Storage, **Canvas**, y otras) son interfaces para librerías incluidas en navegadores. La idea es hacer disponible

poderosas funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la web [15].

2.1.4 Markdown

Markdown es un lenguaje de marcado ligero que trata de conseguir la máxima legibilidad y facilidad de publicación tanto en su forma de entrada como de salida.

El objetivo primordial del diseño de la sintaxis de formato de Markdown es hacerlo lo más legible posible. La idea es que un documento con formato Markdown debería ser publicable tal como está, como texto sin formato, sin parecer que ha sido marcado con etiquetas o instrucciones de formato. Mientras que la sintaxis de Markdown ha sido influenciada por varios filtros text-to-HTML existentes, la fuente más grande de inspiración para la sintaxis de Markdown es el formato de correo electrónico de texto plano [17].

2.1.5 Java

Java es el lenguaje de desarrollo de software más utilizado en todo el mundo. Un objetivo clave de Java es poder escribir programas que se ejecuten en una gran variedad de sistemas computacionales y dispositivos controlados por

computadora. A esto se le conoce algunas veces como “escribir una vez, ejecutar en cualquier parte”. Una característica atractiva de Java es su soporte para gráficos (GUI), el cual permite a los programadores mejorar sus aplicaciones en forma visual [18].

2.1.6 UML

El Lenguaje Unificado de Modelo (UML), es un lenguaje para especificar visualizar, construir y documentar los artefactos de los sistemas software, así como para el modelado del negocio y otros sistemas no software [19].

UML se ha convertido en la notación visual estándar para el modelado orientado a objetos. Comenzó como una iniciativa de Grady Booch y Jim Rumbaugh en 1994 para combinar las notaciones visuales de sus dos populares métodos – los métodos de Booch y OMT (Object Modelling Technique)–, mas tarde se les unió Ivar Jacobson, el creador del método Objectory, y el grupo comenzó a ser conocido como los tres amigos.

UML fue adoptado en 1997 como estándar por la OMG (Object Management Group, organización que promueve estándares para la industria), y continua siendo refinado en nuevas versiones [19].

2.1.7 Modelo Vista Controlador

El patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de frameworks[3] basados en el patrón MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores [20].

Este modelo de arquitectura presenta varias ventajas :

- Separación clara entre los componentes de un programa; lo cual permite su implementación por separado.
- Interfaz de Programación de Aplicaciones API (Aplication Programming Interface) muy bien definida; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.

[3] Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

- Conexión entre el Modelo y sus Vistas dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas. Este escenario contrasta con la aproximación monolítica típica de muchos programas de pequeña y mediana complejidad. Todos tienen un Frame que contiene todos los elementos, un controlador de eventos, un montón de cálculos y la presentación del resultado [19]

2.1.8 Librería

Una librería es un conjunto de funciones de un lenguaje de programación. Con el uso de las funciones que se incluyen en las librerías podemos crear algoritmos ya comprobados y no partir de cero en la creación de los programas [21].

2.2 Marco Biológico

Este subcapítulo es una introducción a la genética para entender algunos conceptos que abordaremos en capítulos posteriores. Aunque cada concepto biológico es un tema en sí, aquí los abordaremos de manera general, ya que sólo pretendemos que se identifique el objeto biológico y su forma de representación gráfica.

2.2.1 Biología

La biología es la ciencia que se encarga del estudio de la materia viva, desde su origen hasta su evolución. Analiza las características y el comportamiento de los organismos individuales, así como de las especies en su conjunto, la reproducción de los seres vivos, de las interacciones entre ellos y el entorno en donde se desenvuelven [22].

2.2.2 Genética

La genética puede definirse simplemente como la manipulación del ADN para estudiar las funciones celulares y de órganos. Dado que el ADN codifica toda la información necesaria para hacer que la célula y el organismo completo, los efectos de cambiar esta molécula puede indicar pistas a las funciones normales de la célula y el organismo.

2.2.3 ¿Qué es una célula?

La célula es la unidad morfológica y funcional de todo ser vivo. De hecho, la célula es el elemento de menor tamaño que puede considerarse vivo. Así que puede clasificarse a los organismos vivos según el número de células que tengan: si solo tienen una, se les denomina unicelulares (como pueden ser los protozoos o las bacterias, organismos microscópicos); si poseen más, se les llama pluricelulares.

También se puede clasificar a las células por su tipo de estructura o composición, es decir la forma en cómo organizan su material genético. Dada esta clasificación las podemos agrupar en células procariotas y eucariotas.

Un procariota o procarionte es un organismo unicelular sin núcleo definido, es decir, cuyo material genético se encuentra disperso en el citoplasma, reunido en una zona denominada nucleoide, por ejemplo arqueas y bacterias. Por el contrario, las células que sí tienen un núcleo diferenciado del citoplasma, se llaman eucariotas, es decir aquellas cuyo ADN se encuentra dentro de un compartimento separado del resto de la célula, por ejemplo animales y vegetales.

En esta tesis sólo nos enfocaremos a los conceptos asociados a la célula procariota, específicamente a la bacteria *Escherichia coli* K12, que es lo que se revisa y anota en RegulonDB.

2.2.4 Genoma

Un genoma es el conjunto completo de ADN de un organismo, incluyendo todos sus genes. Cada genoma contiene toda la información genética necesaria para construir y mantener ese organismo [23].

2.2.5 ¿Qué es el ADN?

El ácido desoxirribonucleico, abreviado como ADN, es el que contiene las instrucciones genéticas usadas en el desarrollo y funcionamiento de todos los organismos vivos conocidos, y es responsable de su transmisión hereditaria.

La información en el ADN se almacena como un código formado por cuatro bases químicas: adenina (A), guanina (G), citosina (C) y timina (T). El orden o secuencia de estas bases determina la información disponible para construir y mantener un organismo, similar a la forma en que las letras del alfabeto aparecen en un cierto orden para formar palabras y oraciones.

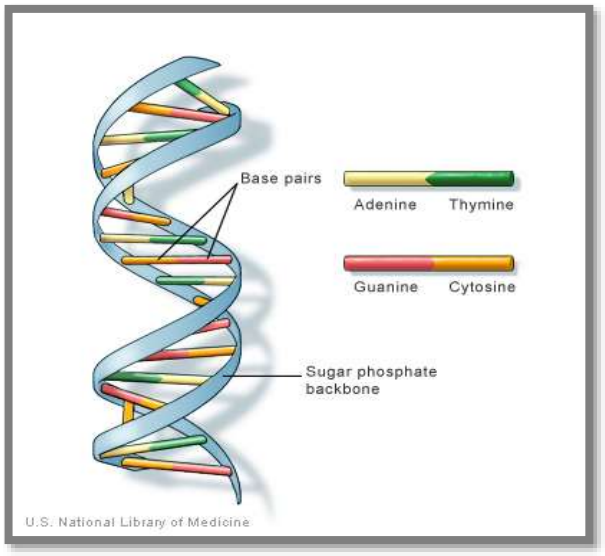


Figura 3: *El ADN es una doble hélice formada por pares de bases unidos a un esqueleto de azúcar-fosfato [24].*

2.2.6 ¿Qué es un cromosoma?

El cromosoma de la bacteria intestinal *Escherichia coli* es circular y contiene cerca de 4.7 millones de pares de bases. Tiene cerca de 1 mm de longitud, pero solo 2 nm de ancho. El cromosoma se replica produciendo una figura que asemeja a la letra griega theta [25].

2.2.7 *Escherichia coli* K-12

Escherichia coli (ver figura 4), originalmente llamada "comuna bacteria coli," fue aislado de las heces de un niño en 1885 por el pediatra austriaco Theodor Escherich (Escherich, 1885). Escherichia coli es un habitante común del tracto gastrointestinal de humanos y animales. Hay cepas de *E. coli* que son comensales inofensivos del tracto intestinal y otros que son los principales agentes patógenos de humanos y animales.

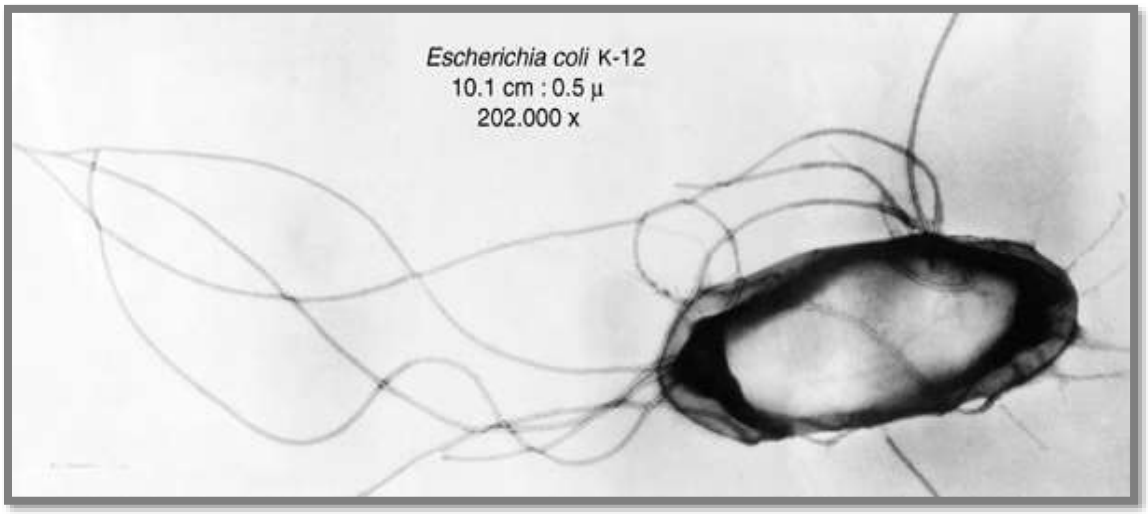
Escherichia coli se puede encontrar en segundo lugar en el suelo y el agua como resultado de la contaminación fecal. Clásicamente, su detección se ha utilizado como un indicador de la mala calidad del agua.

Los procariotas como *Escherichia coli* y los virus que los infectan fueron y siguen constituyendo una herramienta fundamental para el estudio de la estructura y transmisión de los genes.

Entre las ventajas de trabajar con estos organismos encontramos:

- Menor tamaño y mayor número de organismos por área de cultivo
- Mayor velocidad de reproducción: *Escherichia coli* duplica su población en 20´, todos los descendientes son clones de la célula original.
- son haploides, por lo que cualquier cambio o mutación se expresa inmediatamente.

[26]



Electron micrograph of E. coli K-12 by Melvin L Demphilis and Julius Adler [27]

Figura 4:

2.2.8 Regulación Genética

La regulación de la expresión génica depende de la interacción entre el ambiente químico de la célula y las proteínas reguladoras codificadas por genes reguladores. Las proteínas reguladoras pueden funcionar como controles negativos (reprimen el proceso de transcripción) o positivos (aumentan o estimulan la transcripción).

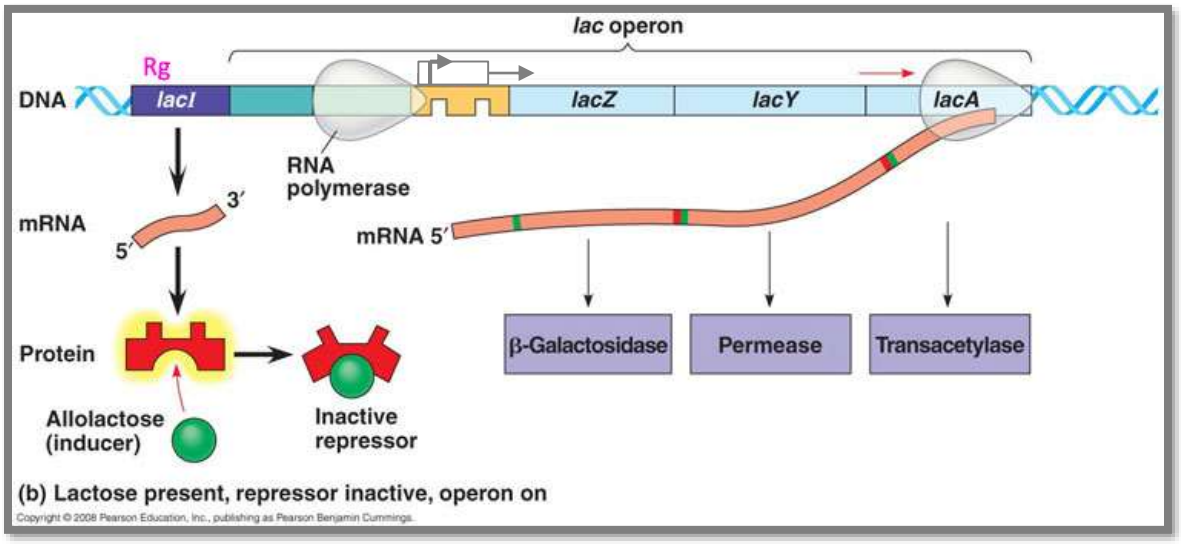
Los procariontes han adquirido la capacidad de utilizar una gran variedad de moléculas para su crecimiento. Por ejemplo, *E. coli* puede fabricar 1,700 enzimas y algunas otras proteínas, además de sintetizar sus aminoácidos a partir de amoniaco y carbono. *E. coli* abastecida de lactosa como fuente de carbono y energía, requiere de la enzima betagalactosidasa para digerirla. Células crecidas

en un medio con lactosa fabrican aproximadamente 3,000 moléculas de la enzima, pero en ausencia de lactosa, hay un promedio de una molécula de enzima por célula. Por lo tanto, la presencia de lactosa induce la síntesis de moléculas de enzima necesarias para degradarla. Se dice que estas enzimas son inducibles.
Las enzimas inducibles son aquellas cuya concentración aumenta en respuesta a señales ambientales.

La presencia de un nutriente determinado puede inhibir la transcripción de un grupo de genes estructurales. Las enzimas cuya síntesis se reduce en presencia de los productos de las reacciones que catalizan, se denominan represibles. Las enzimas represibles o reprimibles son aquellas cuya síntesis se reprime en presencia de productos en reacciones que ellas catalizan [28].

Transcripción en procariotes.

El proceso de transcripción es la síntesis de una molécula de RNAm usando como molde una de las hebras de DNA. Comienza cuando la RNA Polimerasa se acopla al DNA en la región llamada promotor. La RNA Pol de une estrechamente al promotor y hace que la doble hélice del DNA se abra, dando inicio a la transcripción. La cadena de RNA en crecimiento permanece brevemente unida por puentes de hidrógeno al DNA molde (10-12rnt). Posteriormente se desprende como una cadena simple.



Lac operon (Regulation of gene expression in Prokaryotes) [29]

Figura 5:

2.2.8 Operón

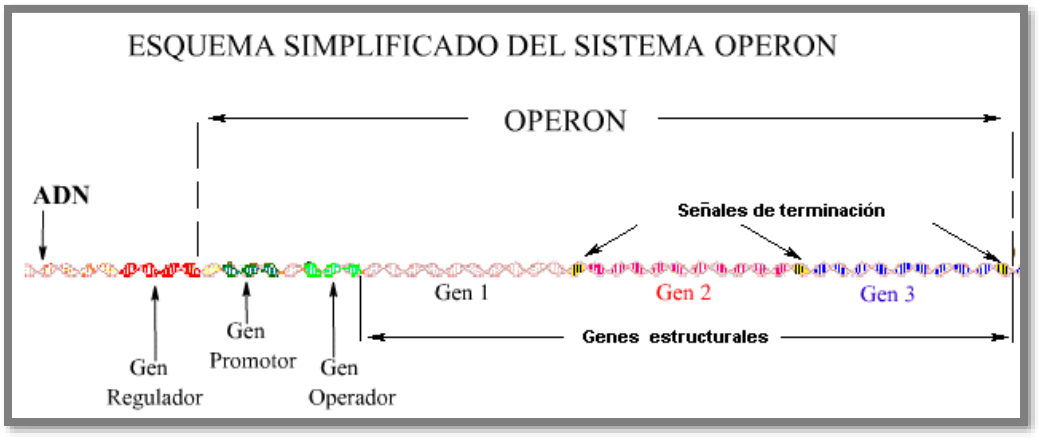
Cabe señalar que el ADN procariota se organiza en paquetes coherentes denominados OPERONES, en los cuales se encuentran los genes para funciones interrelacionadas. El modelo operón de la regulación de los genes procariotas fue propuesto en 1961 por Francois Jacob y Jacques Monod [26].

El operón entonces es una región de DNA que consiste de:

- Un operador: Es una región de dna que controla el acceso a la RNA polimerasa al promotor.

- Un promotor: es la región de dna donde la RNA polimerasa reconoce el sitio donde debe iniciar el inicio de la transcripción.
- Un gen regulador (proteína reguladora): es aquel gene cuyo producto es una proteína que servirá para controlar el tiempo y velocidad de la transcripción de otros genes, a los que se dice que regula.
- Un gene estructural: son el o los genes que forman parte del operon, y que en conjunto se transcriben para generar enzimas que la célula necesitará para la respuesta a una condición de la célula.

El gen regulador codifica para una proteína que se pega al operador, obstruyendo al promotor (y por lo tanto a la transcripción), del gen estructural. El regulador no tiene que estar adyacente a los otros genes en el operón. Cuando se remueve la proteína represora, puede producirse la transcripción. El operador y el promotor son sitios de unión sobre el ADN y no se transcriben [26].

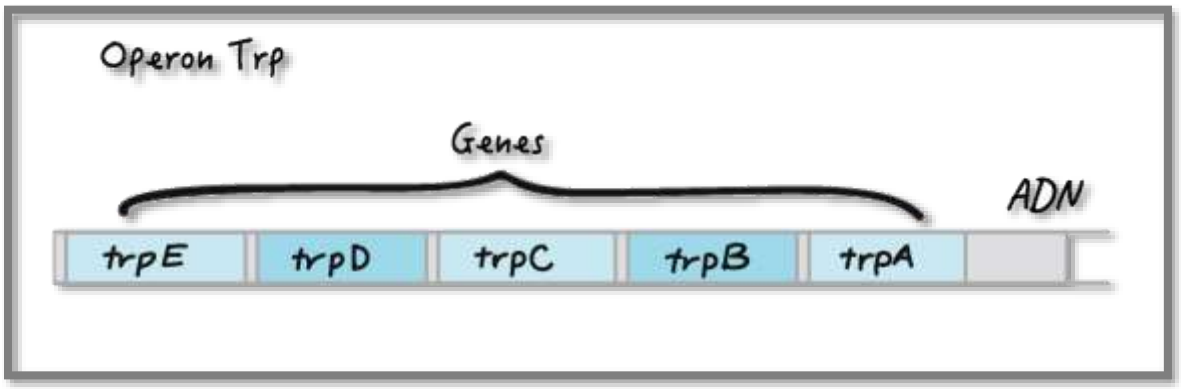


Representación de un Gen en el ADN.

Figura 6:

2.2.9 Gen

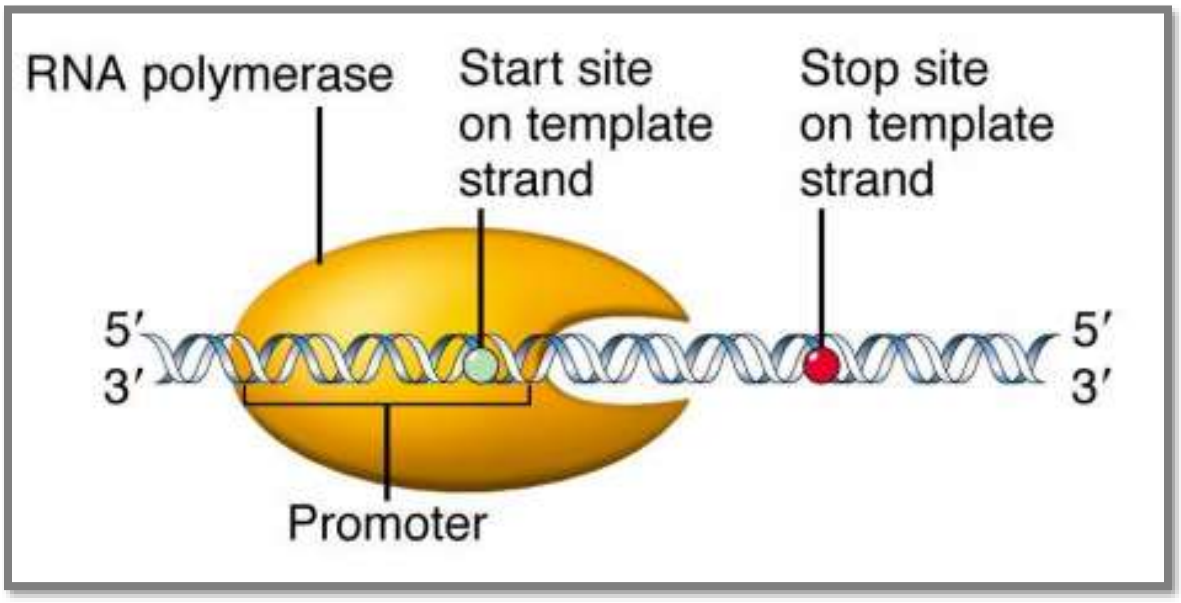
Un gen es la unidad básica o fundamental de la herencia, y podemos describirlo como el segmento de ADN implicado en la producción de una cadena de polipéptido o ARN estable [30], ver figura 7.



Representación de un Gen en el ADN.

2.2.10 Promotor

Un promotor es la secuencia de ADN, donde la ARN polimerasa se une e inicia la transcripción. Notas: La región promotora donde se une la polimerasa es una región con un patrón específico para que los diferentes factores sigma, que es parte de las proteínas que componen la polimerasa, reconozcan esa región de DNA y se unan. Un promotor se representa como un región de 60 pares de bases arriba y 20 pares de bases abajo de donde se inicia el proceso de transcripción, también llamado +1, [30]; ver figura 8.



Promoter [31]

2.2.11 TFs binding sites

Los sitios de unión de TFS son sitios de ADN físicos reconocidos por factores de transcripción dentro de un genoma. Nota: Históricamente, los sitios de unión para reguladores transcripcionales se definen como sitios operadores. Hay varios significados de un sitio de operador. Sitios operadores en su significado más amplio son sitios para represores o activadores. Más adelante, el término "activator" sitios se oponen al término "activator sites", que se limitó a sitios de unión para los reguladores de represor [30]; ver figura 9.

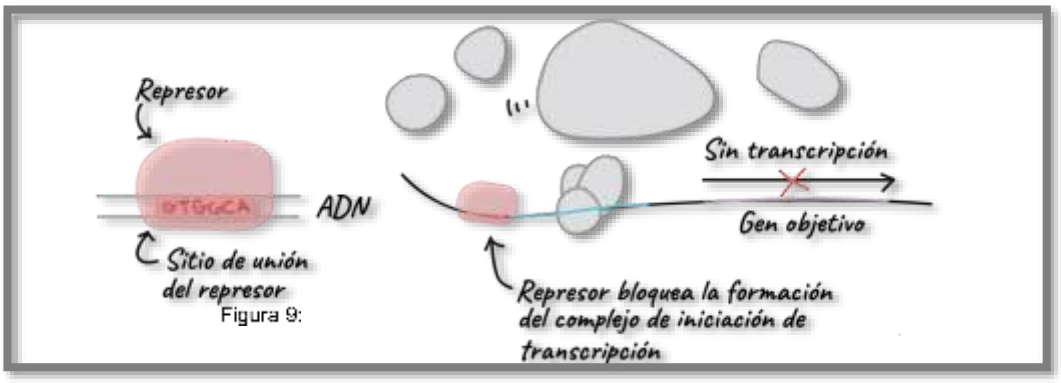


Figura 9:

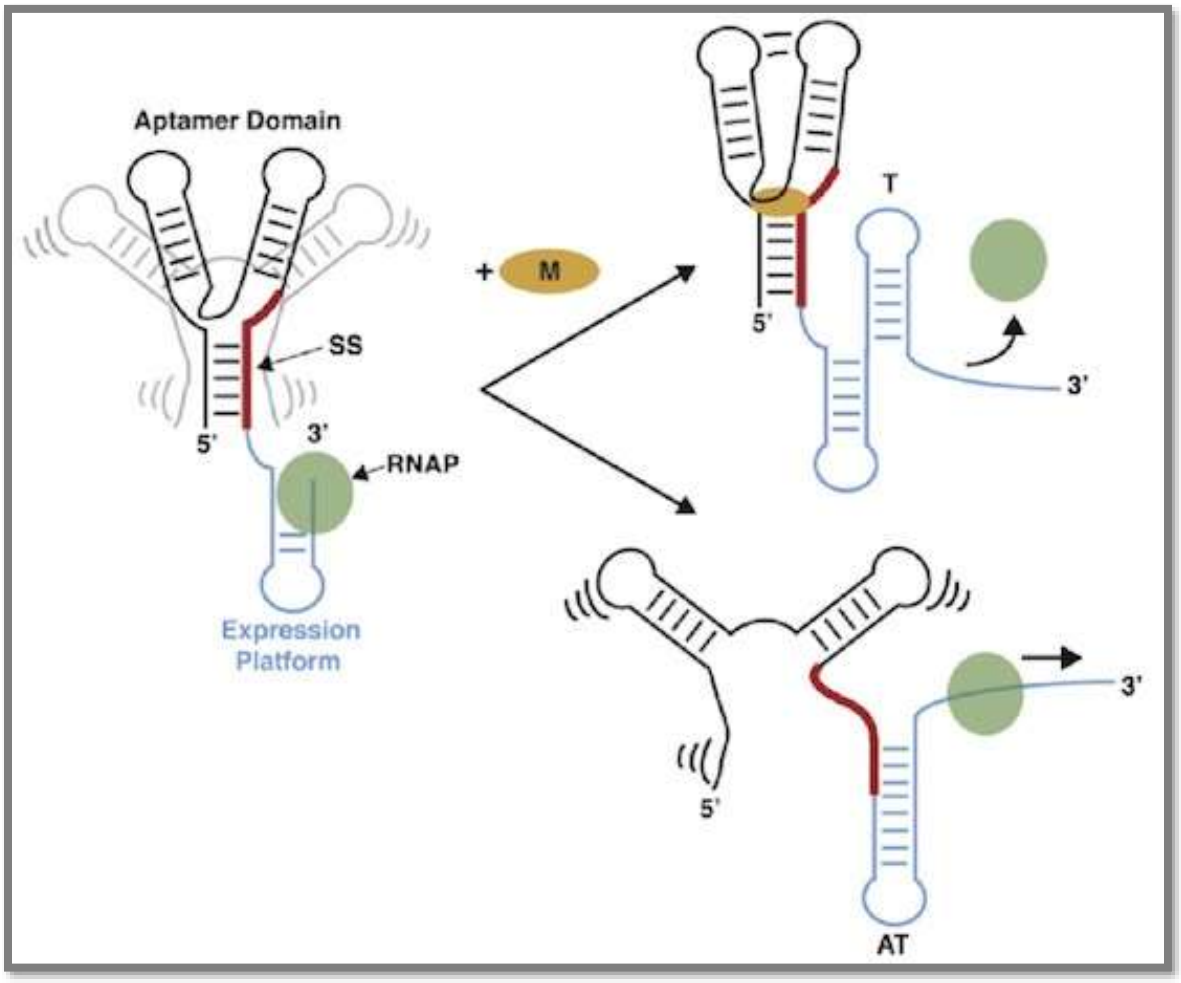
TF binding site [32]

2.2.12 Riboswitch

Un riboswitch es un segmento de una molécula de ARN mensajero que tiene un efecto regulatorio y al que se le une una molécula pequeña, resultando en un cambio en la producción de las proteínas codificadas por el mRNA.

Los riboswitches son importantes en las bacterias debido a que es una manera a través de la cual han ido perfeccionando su habilidad de poder regular la producción de recursos a través de este método de regulación de la expresión génica que permite disminuir la pérdida de recursos [33].

La molécula de RNA mensajero se pliega formando estructuras que son otra forma de regulación genética.



Riboswitch in Genome [34]

2.2.13 Atenuador

El atenuador, región que se compone de secuencias encuentran en el ARN transcrito, están implicadas en el control de la transcripción de operón después de la ARN polimerasa ha iniciado la síntesis. El atenuador de las secuencias de ARN

se encuentran cerca de la 5 ' de la ARN llamado el líder de la región de ARN. La secuencia líder está situada antes del inicio de la codificación de la región para el primer gen de la Operón [30].

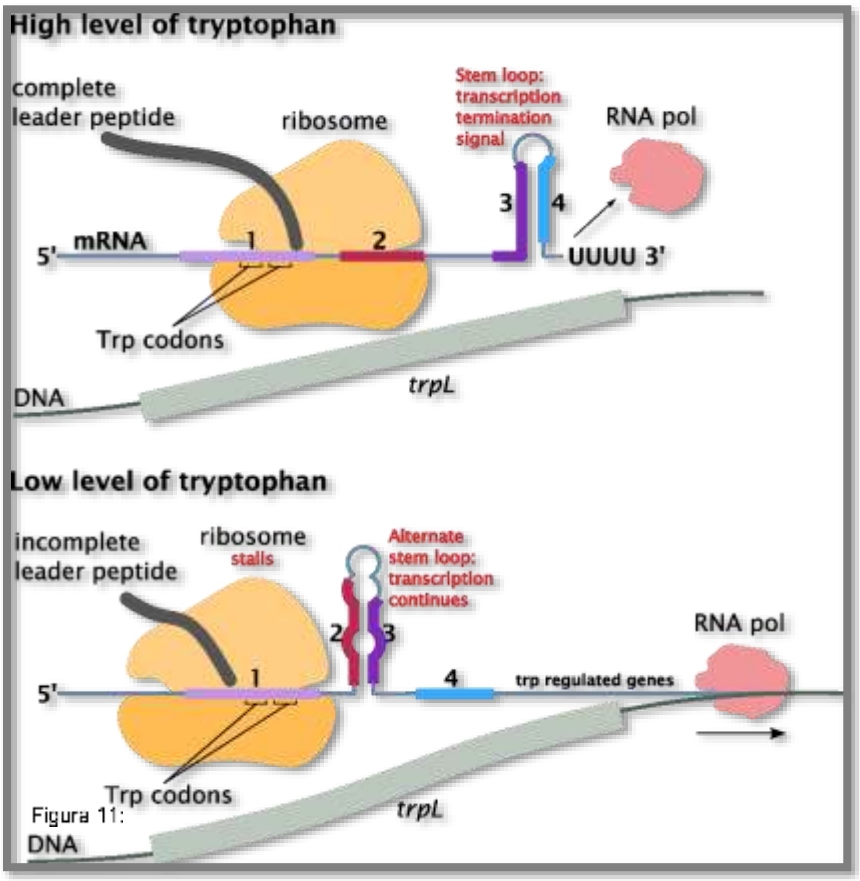


Figura 11:

Trp operon attenuation [35]

2.2.14 Terminador

Conjunto de secuencias que marcan el fin de la transcripción de una unidad de transcripción [30].

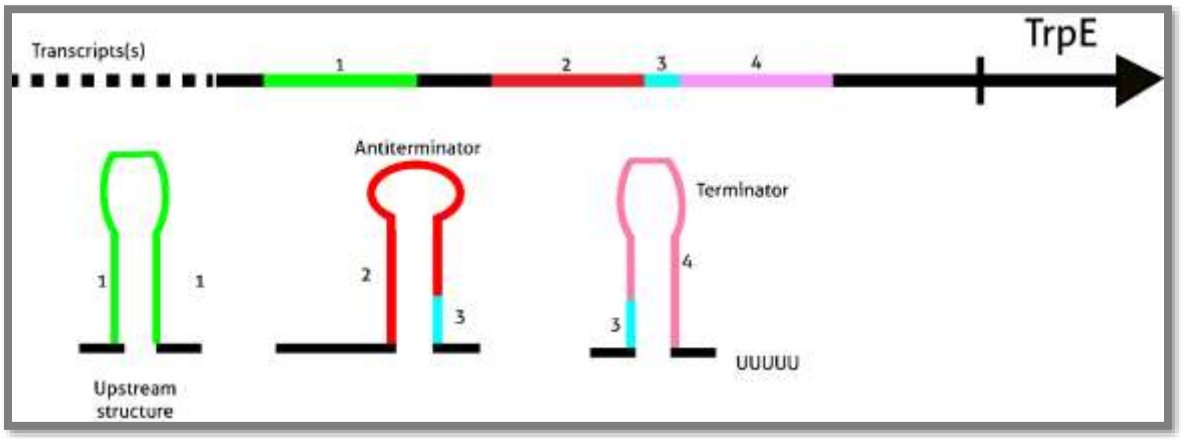


Figura 12: *Los diferentes roles del ARN de transferencia de triptófano en la regulación [36]*

2.2.15 Small RNAs

Pequeños RNAs (sRNAs) son ARN que tienen un papel regulador en la expresión génica. Estos ARN son pequeños, alrededor de 350 nucleótidos y no se traducen en proteínas. Este tipo de ARN fue nombrado miARN en eucariotas y sRNA en bacterias, [30].

2.2.16 Allosteric regulation of RNAP

Una alarmona es una molécula de señalización intracelular en algunas bacterias y plantas. Se producen debido a factores ambientales ásperos y regulan la expresión génica a nivel de transcripción. Entre las causas de la producción de estas moléculas se encuentra la falta de metabolitos como glucosa, la falta de aminoácidos y la presencia de productos metabólicos tóxicos (alcohol, por ejemplo). Algunos factores estrictos toman ARNt no cargado y lo convierten en una alarmona. La alarmona arquetípica es ppGpp (3'-difosfato,5'-difosfato Guanosina), ppGpp se une a las subunidaddes β y β' de la ARN polimerasa, modificando la afinidad por promotores. La transcripción de ARNm se modifica y aumenta la transcripción de genes implicados en la síntesis de aminoácidos.

La proteína DksA coopera con ppGpp de responder al nivel de ppGpp para regular la expresión de genes particulares.[30]

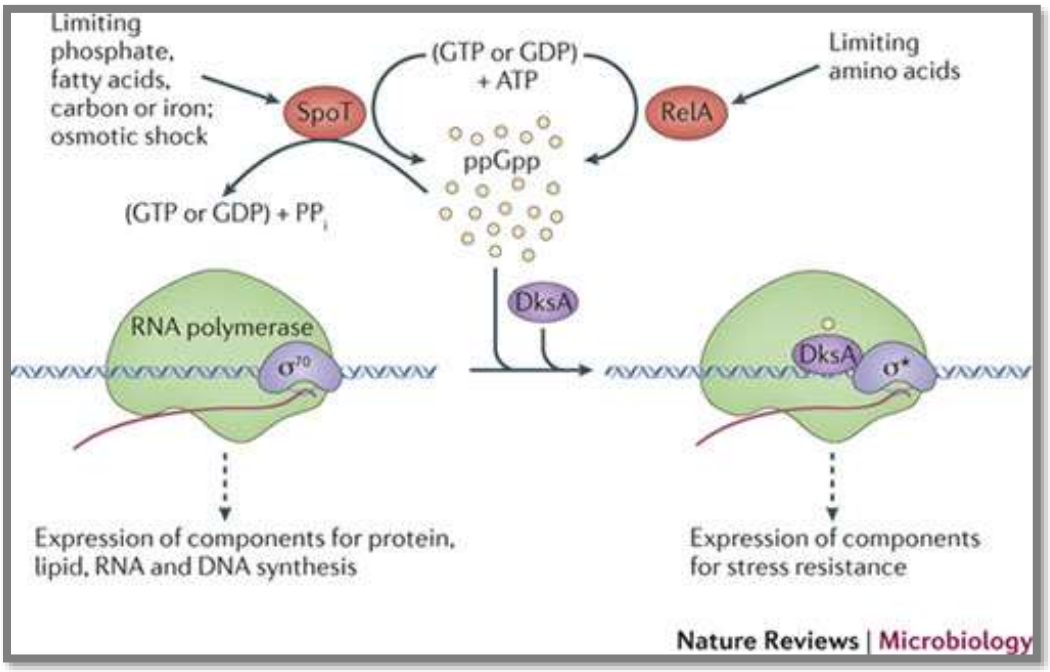


Figura 13: *Representacion de ppGpp [37]*

CAPÍTULO 3:

ANALISIS DE REQUERIMIENTOS

La captura y el análisis de los requerimientos del sistema es una de las fases más importantes para que el proyecto tenga éxito. Como regla de modo empírico, el costo de reparar un error se incrementa en un factor de diez de una fase de desarrollo a la siguiente, por lo tanto, la preparación de una especificación adecuada de requerimientos reduce los costos y el riesgo general asociado con el desarrollo [38]. Los requerimientos de éste proyecto están definidos en el documento de Especificación de Requisitos, además la documentación de la versión 2.0 de la herramienta DrawingTracesTool cuenta con su propio manual, por lo que nos enfocaremos sólo a describir las modificaciones que será necesario realizar.

Como hemos mencionado en capítulos anteriores, la representación gráfica en biología es crucial para el entendimiento de los mecanismos de la célula, y debido a que esos procesos son imperceptibles a la vista humana, con mucho más razón es necesario “dibujar” lo que ocurre en la célula.

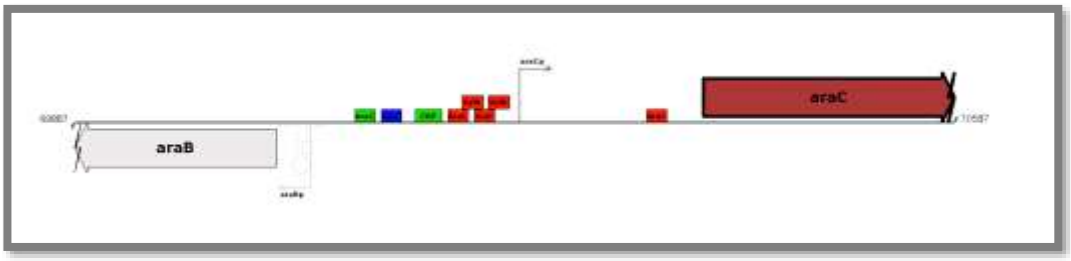
Por otro lado la curación de artículos científicos es una tarea importante para entender el funcionamiento a través de los experimentos que se hacen y son publicados en los artículos científicos, y en varios de ellos “dibujan” o representan los mecanismos o descubrimientos del comportamiento de la célula que han realizado.

Para complementar el conocimiento descriptivo, en RegulonDB se ha desarrollado una herramienta llamada DrawingTracesTool que permite generar imágenes sobre la regulación genética, que es el tema al que se enfoca RegulonDB.

La herramienta fué desarrollada en Java y actualmente genera todos los archivos gráficos que RegulonDB despliega en su interfaz web en los resultados de una búsqueda de genes.

3.1 Drawing Traces Tool (DTT) versión 2.0

DTT permite representar cada uno de los elementos genéticos que se encuentran almacenados en la base de datos de RegulonDB y presenta al usuario el contexto genómico de los genes de manera gráfica, como se ve en la figura.



Representación gráfica del gene AraC en RegulonDB

Internamente, DTT está compuesto por distintos paquetes que entre sus funciones principales se encuentra la lectura y validación de datos fuente, así como la organización de los elementos a graficar. Además, hace uso distintas librerías para graficar los diferentes elementos genéticos. Esta herramienta se ejecuta vía consola y recibe como entrada un archivo de datos de tipo texto (TXT), en estos archivos se describen los atributos de los elementos biológicos como: Gene, Operón, Sitios de unión, Promotor, Terminador, Atenuadores, Riboswitch y Small RNA y la regulación por el elemento ppGpp.

La herramienta DTT puede generar como resultado una imagen en formato png/jpg además de un archivo HTML que integra la imagen y el despliegue de tooltips. Aunque la distribución del DTT se hace por medio de un archivo .jar, internamente refiriéndose a código fuente, la herramienta está compuesta por un proyecto con tres paquetes y sus respectivas clases. Además, el DTT integra como librería la API de graficado llamada RegulonDBGraphics (en formato .jar).

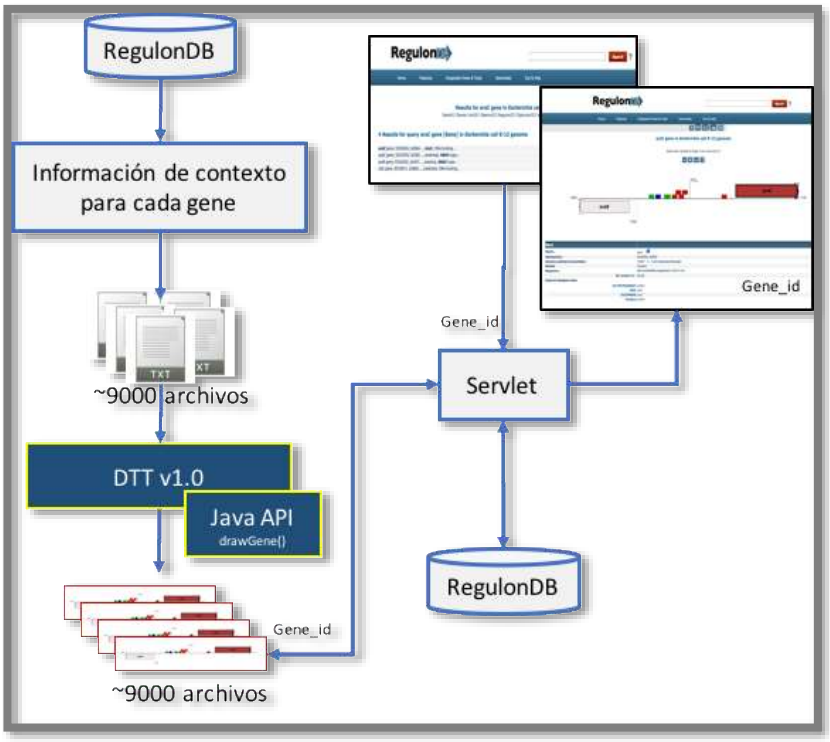


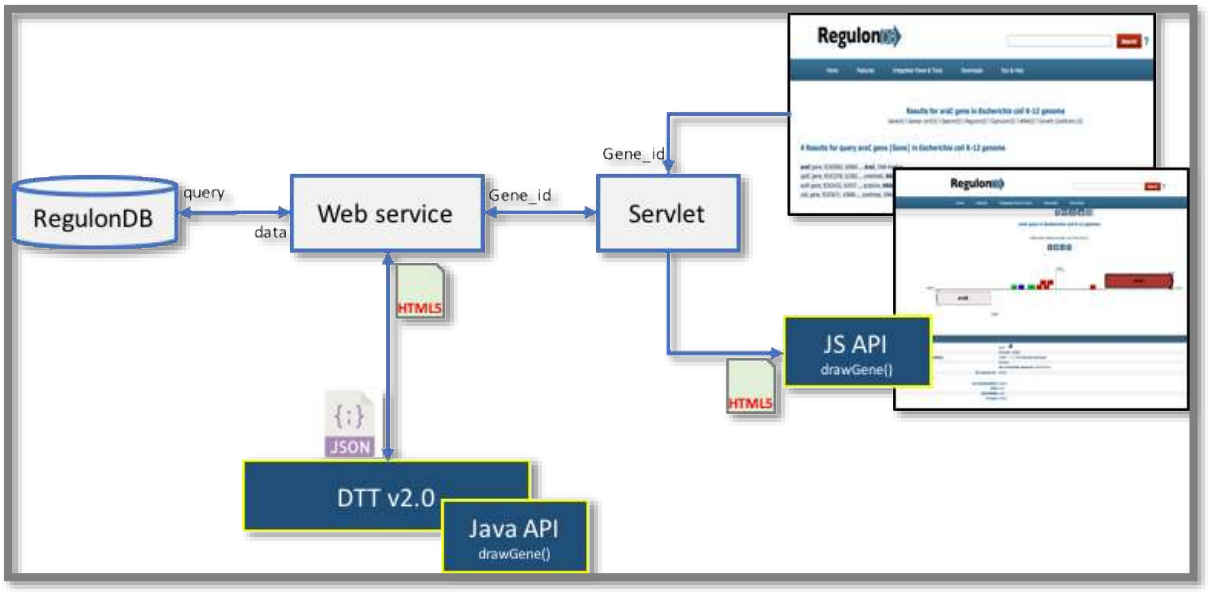
Figura 15: *Esquema general de la aplicación DTT v1.0*

Lo que se busca con el desarrollo de este proyecto es que por medio de las imágenes que se generen a través de DTT el usuario pueda interactuar con las ellas y como beneficio adicional realizar la integración de los resultados del DTT con RegulonDB. De esta forma el DTT realizará el trabajo completo de organización y creación de las imágenes dinámicas y ese resultado será incorporado a RegulonDB. Logrando con esto que el DTT funcione de manera independiente y pueda ser integrado en cualquier sistema que requiera de sus funcionalidades.

3.2 Requisitos Funcionales

Dada la revisión que se hizo a la versión 1.0 del DTT, se lograron obtener los requisitos a implementar. La figura mostrada abajo, es el esquema de componentes del proyecto.

La interfaz web, específicamente el servlet encargado de desplegar los resultados de un gene, es el que debe ejecutar el servicio web. A éste se le pasa el identificador del gene, y con él accede a la información del contexto del gene. Un archivo JSON es generado con los resultados de la consulta a la base de datos, y es la entrada para que el DDT pueda validar, organizar y crear el archivo de instrucciones que el API en JavaScript necesita para graficar. El API es el que se tiene que implementar, haciendo que las funcionalidades que tiene el API en Java existan las mismas funcionalidades en el API de JavaScript. Por lo que podemos decir que el API Java y el API de JS deben ser librerías homólogas.



Esquema general del proyecto.

Figura 16:

Tabla 1: Lista de requisitos.

Requisito	***Nombre del requisito***
R1	Crear la versión 2.0 del DTT que permita generar archivos en formato HTML5 con las instrucciones para graficar en JS.
R2	Crear el API de JavaScript con funciones homólogas a las del API de Java.
R3	Integrar las herramientas en RegulonDB para el despliegue de las imágenes de manera dinámica en la página de resultados de genes.

R1. Crear la versión 2.0 del DTT

Actualmente la versión 1.0 del DTT permite crear imágenes jpg/png o bien archivos html con la imagen integrada pero estática. Lo que se pretende con ésta nueva versión es hacer que el DTT genere archivos de salida en formato HTML5 que contengan instrucciones JavaScript del API de graficación para pintar la información del contexto del gene.

Esta versión también se actualizará para poder leer archivos JSON como entrada, eliminando la funcionalidad de lectura de archivos XML que dejó de mantenerse.

R2. Crear librería gráfica en JavaScript

Para la creación de la librería gráfica homóloga en Javascript deberán crearse todas las funcionalidades de manera idéntica, de tal forma que no exista diferencia entre el API de Java y el de Javascript. Los elementos gráficos que actualmente existen son los que muestra la figura 17.

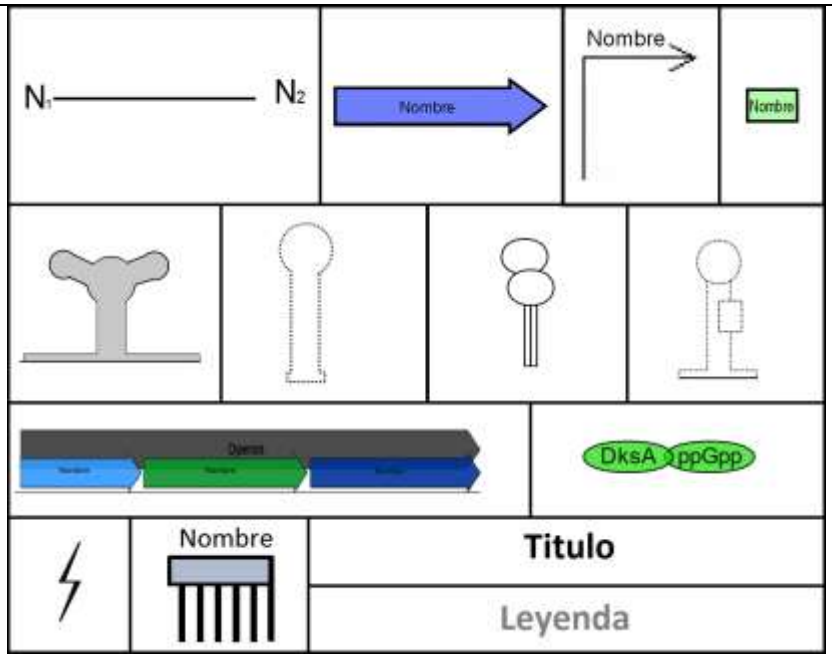


Figura 17: *Elementos que participan en la regulación transcripcional*

Deberán integrarse todas las propiedades permitidas a los gráficos:

- Posición izquierda y derecha de cada objeto o sólo izquierda para el caso del promotor
- Altura.
- Orientación del objeto, si el strand del elemento es *forward* el elemento apunta a la derecha, si es *reverse* apunta a la izquierda.
- Los elementos pueden o no tener una etiqueta que contendrá el nombre de elemento.
- El tipo de fuente de las letras hay un default, pero también puede indicarse en el archivo de entrada.
- Las etiquetas no siguen la orientación de los objetos, siempre van de izquierda a derecha, nunca de cabeza.

- El contorno de los elementos puede modificarse, especificándolo en el archivo de datos.
- El tipo de línea puede modificarse, especificándolo en el archivo de datos.
- Si lo desea, el usuario puede graficar elementos “incompletos” (ver figura 18.

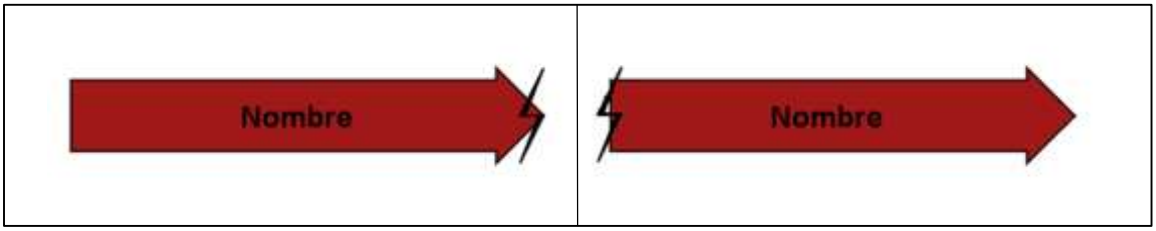


Figura 18: *Ejemplo de elementos incompletos.*

El API debe tener un manual de usuario escrito en MarkDown.

R3. Integración de las herramientas en RegulonDB

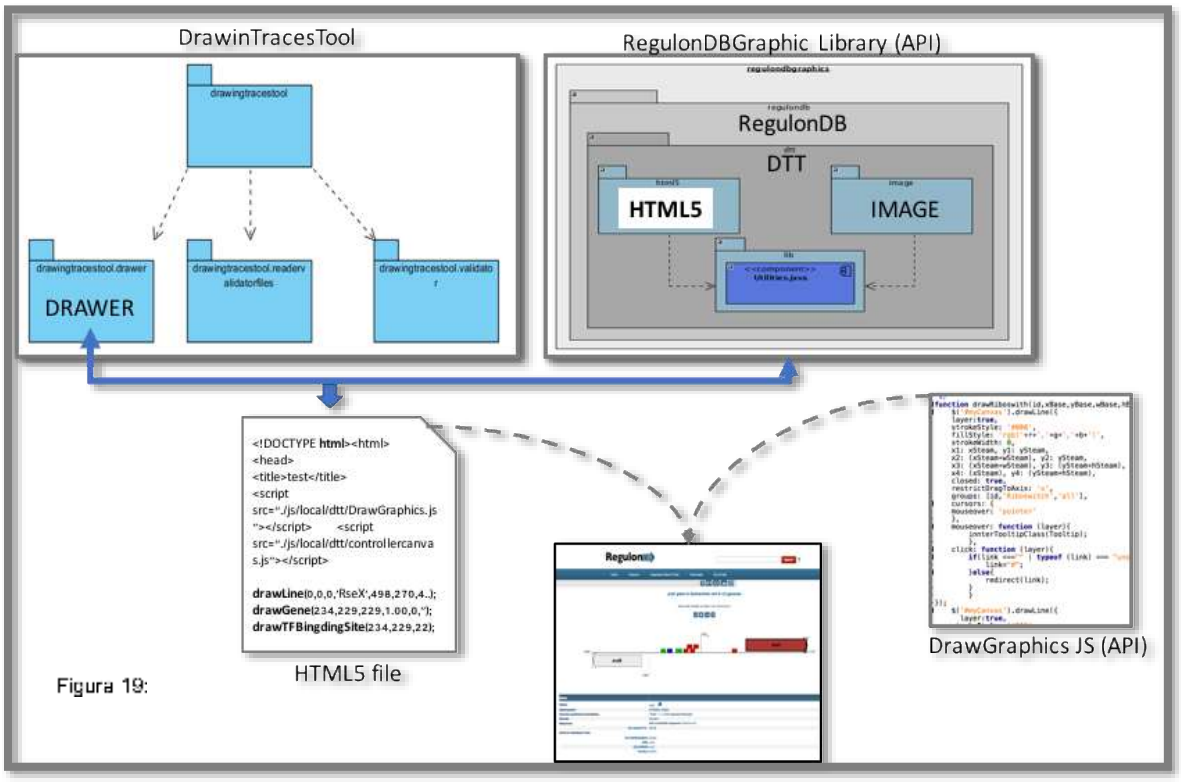
La librería gráfica de JavaScript será importada al navegador web que contendrá el elemento <canvas> de HTML, al cual se asignaran las representaciones genéticas que sean invocadas.

Las funcionalidades extras a los elementos genéticos como lo son: help, zoom in, zoom out, expand, save, screen y restore; estarán en un archivo ajeno al API (**controllercanvas.js**), esto como mejora de desarrollo de software.

CAPÍTULO 4:

DISEÑO DEL PROYECTO

El diseño final del proyecto se muestra en la siguiente figura, donde podemos ver la interacción entre el DTT, la librería gráfica RegulonDBGraphics y el API de JavaScript.



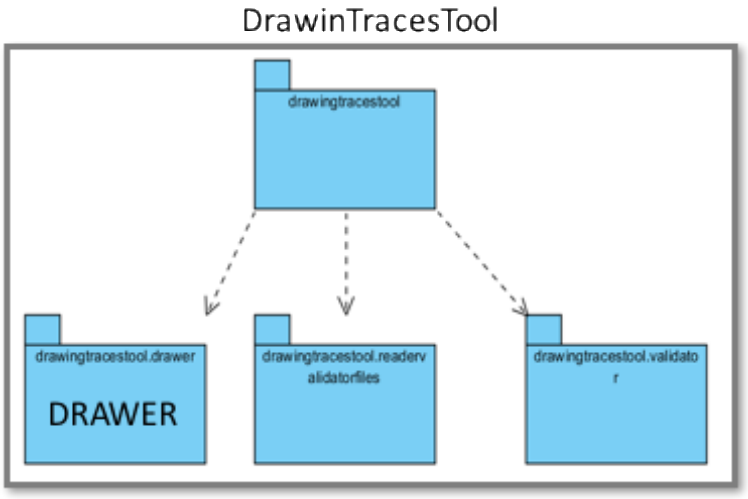
Esquema General de diseño de los componentes del sistema.

El DTT permite recibir la opción de salida HTML5 desde línea de comandos, y el API RegulonDBGraphics en Java tiene un módulo para generar el archivo HTML5, que contiene instrucciones tipo JavaScript para ser interpretados posteriormente por el API de JavaScript.

4.1 Crear la versión 2.0 del DrawingTracesTool

Mencionábamos en el capítulo anterior, que es necesario generar la segunda versión de la aplicación integrando la funcionalidad de permitir la creación de archivos en formato HTML5 conteniendo las instrucciones del API de Javascript para la graficación de los elementos genéticos.

La versión 1.0 realizaba la lectura y validación de archivos, valida parámetros, dibuja imágenes (jpg/png) y ahora en la versión 2.0 tiene esas mismas funcionalidades más la creación de archivos html5.



Componentes del DTT 1.0

Figura 20:

La aplicación contiene un subpaquete ***Drawer*** donde están las clases encargadas de dibujar los elementos de regulación genética, ordenarlos y organizarlos; las clases son: `Drawer.java`, `DrawerHTML5.java`, `SortingMethod.java`, `OrganizerMethod.java`.

La clase implementada para esta versión fue el DrawerHTML5.java, que puede verse en la figura 21.

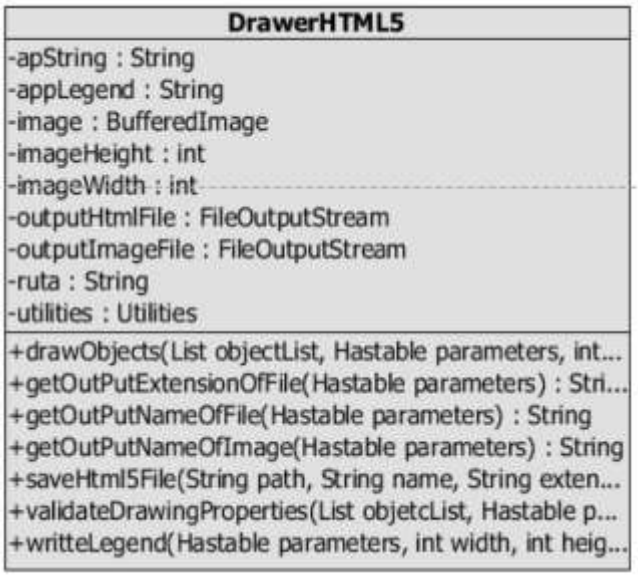


Diagrama de Clases del subpaquete DrawingTracesTool.drawer

Figura 21:

DrawerHTML5.java: Esta clase es la encargada de dibujar los objetos correspondientes de la lista y el establecimiento de las posiciones que se deben extraer, dedicada exclusivamente para trabajar en el *lienzo* de html5.

4.2 Diseño de la librería gráfica en Java

El diseño de la librería gráfica RegulonDBGraphic quedó organizada de forma que el paquete principal es RegulonDB, el subpaquete es DTT, el cual contiene diferentes clases para el manejo de las figuras en varios formatos, por el

momento incluye la generación de los gráficos en imagen o bien en código HTML que contiene instrucciones de Javascript, como se observa en la figura 19.

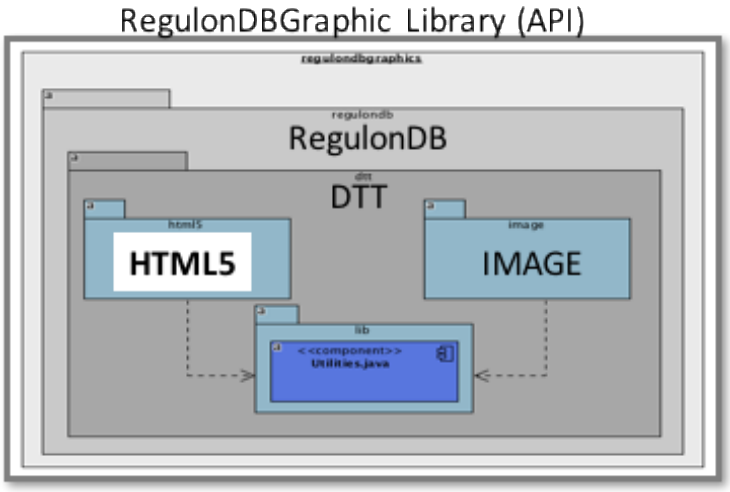


Figura 22: *Diseño del paquete RegulonDBGraphics*

Organizar de esta manera las clases, permitió también organizar una librería de Utilities que contiene métodos que son comunes a ambas clases.

La clase HTML5 fue creada en su totalidad, y es una clase homóloga a la de Imagen, conteniendo los mismos métodos, tanto en nombre como en los parámetros, la diferencia estriba en la salida que genera.

La figura 20 muestra los métodos de la clase HTML5.

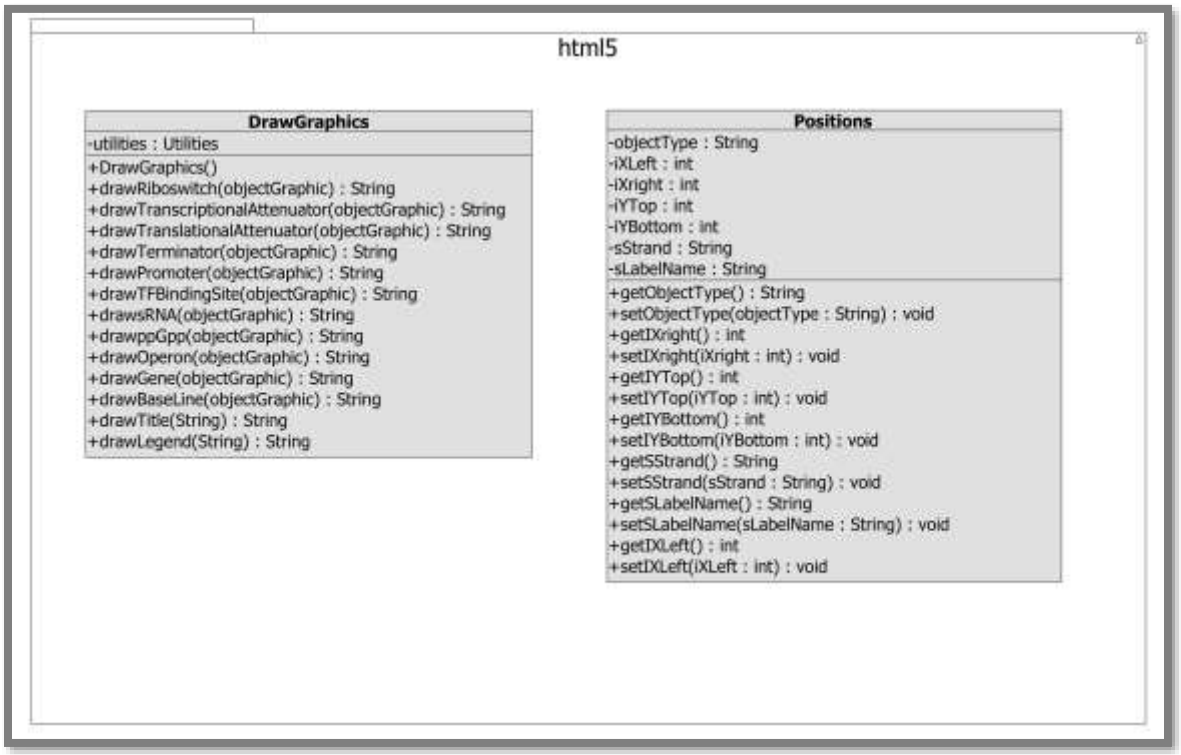


Figura 23: *Diagrama de clases del paquete html5 DrawingTracesTool.html5*

1. **Positions.java**: Clase donde se almacenan todas las posiciones, sentido, etiqueta del elemento a dibujar, necesarias para la clase DrawGraphics.
2. **DrawGraphics.java:** Esta clase crea todas las instrucciones en JavaScript de cada elemento gráfico, que serán usados por los métodos de la librería llamada *DrawGraphics.js, figura 24*.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Ejemplo Tesis</title>
</head>
<script src="./js/local/dtt/DrawGraphics.js"></script>
<script src="./js/local/dtt/controllercanvas.js"></script>
<script type="text/javascript">
    drawLine(0,0,0,'RseX',498,270,4,'...');
    drawGene(234,229,229,1.0,0);
    drawTFBingdingSite(234,29,2);
</script>
<body>
</body>
</html>
```

Ejemplo de salida funciones javascript.

Figura 24:

4.3 Diseño de API DrawGraphics.js

El diseño de la librería en JavaScript utilizara los mismos nombres de los métodos que en Java y se agregaran algunos más para dibujar la marca de agua. Esta API no tendrá clases, simplemente funciones que generan las figuras directamente en el área del canvas y ya que cada función dibuja en base a los parámetros que se le establecen puede utilizar las veces que sean necesarias la función y dibujar N veces la misma figura si se requiere. En la siguiente figura 22 podemos observar el diagrama de clases del api.

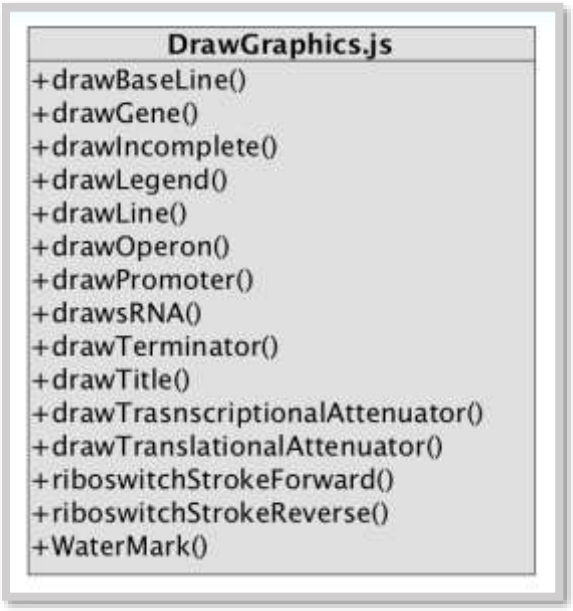


Diagrama de clases DrawGraphics.js

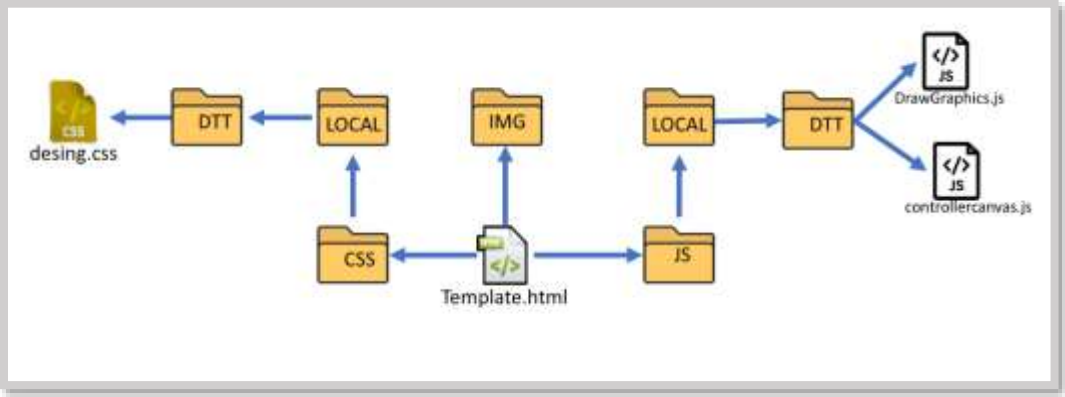
Figura 25:

4.4 Diseño de la estructura HTML

Tomando en cuenta las especificaciones establecidas en los temas anteriores se comenzó con la maquetación del archivo de salida (template), que serán los archivos utilizados principalmente por RegulonDB; dicha maquetación se basó en la estructura recomendada para el diseño de páginas web, separar los archivos por tipo, ver figura 26:

1. CSS: Carpeta para hojas de estilo, dentro de esta existen otros archivos .css que pertenecen a RegulonDB, los archivos para este proyecto se ubican en: ccs/local/dtt/design.css.

2. JS: Carpeta para librerías y código JavaScript, al igual que css RegulonDB utiliza diferentes librerías js, los archivos JavaScript creados para este proyecto se ubican en: js/local/dtt/*.js.
3. IMG: Carpeta para las imágenes clasificadas con la misma estructura de directorios, las imágenes utilizadas en este proyecto se ubican en: images/dtt/*.png.
4. /: El directorio raíz contendrá los archivos HTML.

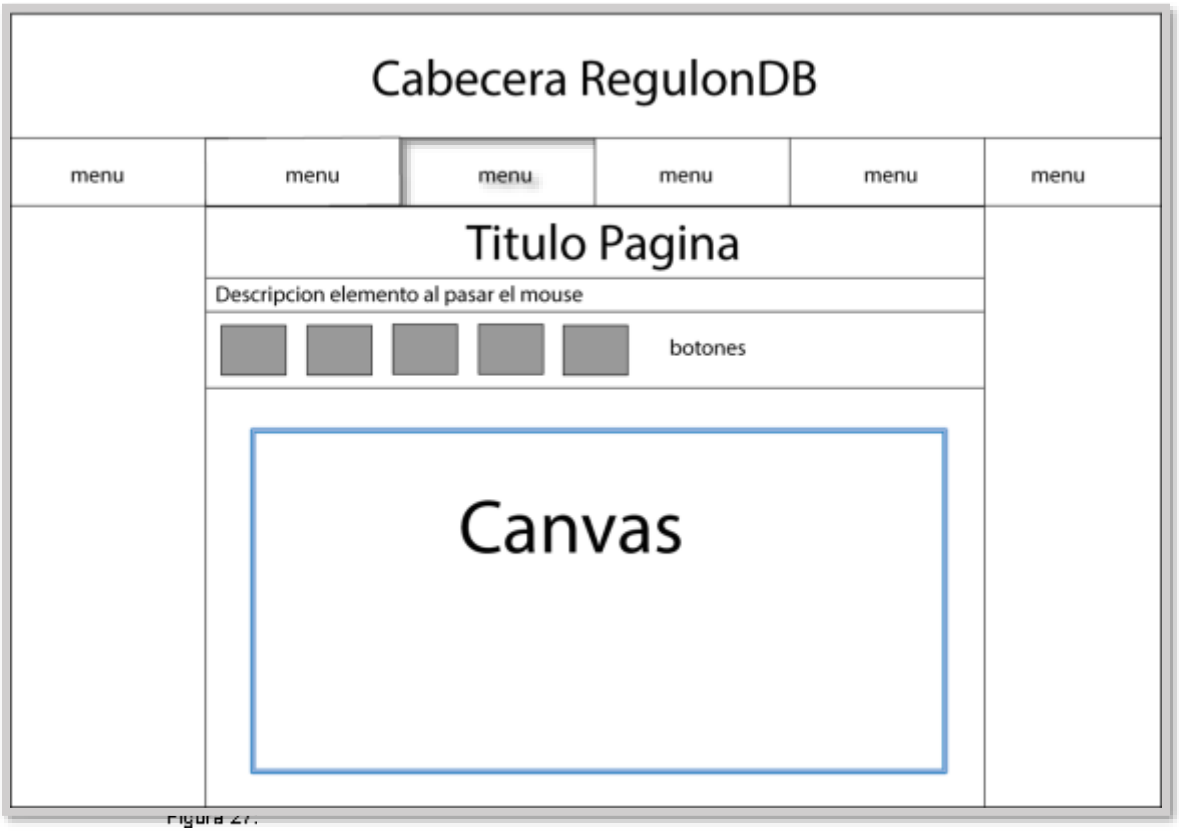


Estructura web.

4.5 Maquetado para Template.html

El desarrollo del archivo HTML fue relativamente sencillo debido a que el uso será un *frame* dentro de otra página, sin embargo, eso no quiere decir que no se cumpla con las etiquetas correspondientes a la tecnología html5. A

continuación, se muestra la parte visual del navegador compuesta de 3 partes indispensables para cumplir con las metas establecidas.



Maquetado de página en RegulonDB.

4.6 Despliegue de la información en la página web.

La forma en que se despliega la información en la página de RegulonDB es en 4 niveles Drill Down ordenados de la siguiente manera:

1. **Drill Down nivel 1:** Se consultará a la base de datos por gen y su región reguladora en ambos sentidos del gen.
2. **Drill Down nivel 2:** Un *web-service* regresara la información a la página de gene para proporcionársela a la herramienta *DrawingTracesTool*.
3. **Drill Down nivel 3:** Se ejecutará la herramienta y la pagina espera el resultado de la herramienta, dado que la salida es un archivo HTML que es incrustado en el *frame* de RegulonDB.
4. **Drill Down nivel 4:** El código incrustado por la herramienta es interpretado por el lenguaje JavaScript mostrando la visualización de la imagen.

CAPITULO 5

DESARROLLO DEL API

5.1 Mantenimiento al API v1.0

Una vez que se obtuvo todo el diseño para desarrollar el API v2.0 se tomó la decisión de trabajar en modo *stand-alone* vía línea de comandos y no afectar el servidor web.

Lo primero que se tuvo que hacer es obtener la versión previa del Drawing Traces Tools y ejecutarla en la maquina local, para no sufrir mucha discrepancia

entre las versiones de la librería se desarrolló el API en la herramienta NetBeans 8.2; en primera instancia se tuvo que dar mantenimiento a esta API ya que utilizaba clases obsoletas de Solaris las cuales fueron removidas.

Con esto ya se tiene listo el API v1.0 lista para desarrollar los módulos especificados anteriormente en los capítulos 3 y 4 de esta tesis.

5.2 Herramientas y espacio de trabajo.

Las herramientas utilizadas se instalaron un grupo de herramientas para desarrollo en *front-end* y *back-end* como lo son:

- **Java 8:** Se instaló este lenguaje de programación para el desarrollo.
- **NetBeans 8.2:** IDE en el que fue desarrollada la versión 2.0 del Drawing Traces Tools y seguir dando mantenimiento al API.
- **Brackets 1.7:** Para desarrollo y maquetación del ámbito web.
- **Visual Paradigm:** Desarrollar los diagramas de clase y de actividad.
- **RStudio:** Herramienta para desarrollo de los manuales y documentación correspondiente.
- **Illustrator: Iconos y maquetado de la página en RegulonDB.**

Una vez listo el ambiente de trabajo, se creó un espacio de trabajo dentro de Dropbox para llevar el desarrollo y mantenimiento de software como lo

establece MoProSoft, con las plantillas y estructura adecuadas al Programa de Genómica Computacional, ver figura 25.

Figura 28: *Espacio de trabajo en Dropbox.*

5.3 Desarrollo del paquete html5 en RegulonDBGraphics

Se creó un paquete nuevo para seguir manteniendo la modularidad y permitir actualizaciones a futuro, dicho paquete contiene las clases desarrolladas para completar las actividades designadas a este módulo.

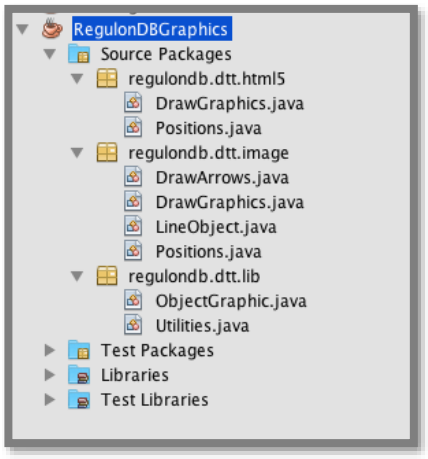


Diagrama de árbol de RegulonDBGraphics.

Figura 29:

Una vez desarrollado la estructura del módulo se crearon las clases DrawGraphics y Positions con las cabeceras de documentación necesarias establecidas en el Programa de Genómica Computacional de CCG; a continuación, se muestra la cabecera de la Clase DrawGraphics (ver figura 30) y Positions (ver figura 31).

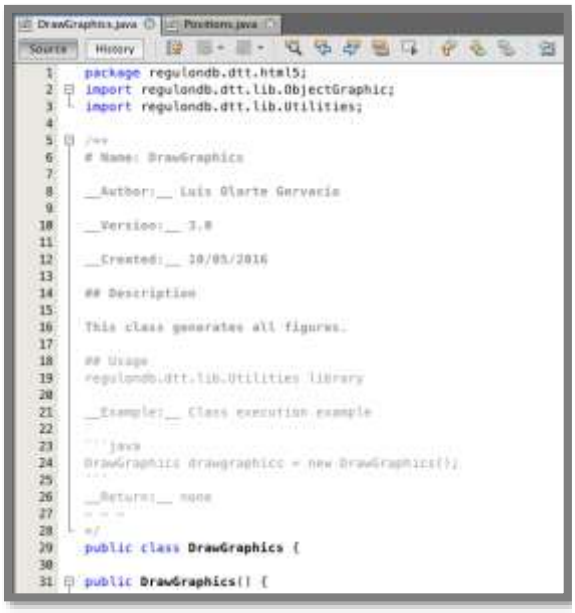


Figura 30: *Cabecera Clase DrawGraphics.java*

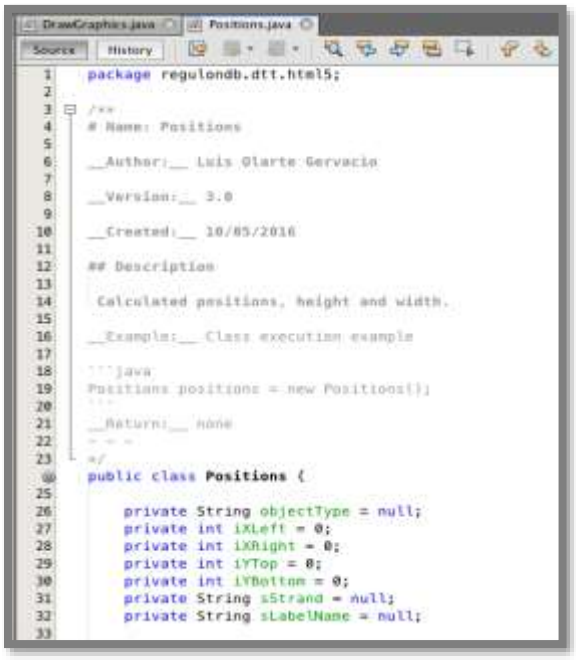


Figura 31: *Cabecera Clase Positions.java*

Cuando las Clases fueron desarrolladas se construyó otra Clase llamada DrawerHTML5 capaz de controlar el API desde la herramienta Drawing Traces Tools.

5.4 Desarrollo Clase DrawerHTML5 en Drawing Traces Tools

La herramienta Drawing Traces Tools estructurada de la siguiente manera (ver figura 32); dentro del paquete drawer se desarrolló DrawerHTML5.java, esta al mismo nivel de la clase anterior la Drawer.java la cual crea imágenes en formato PNG o JPG.

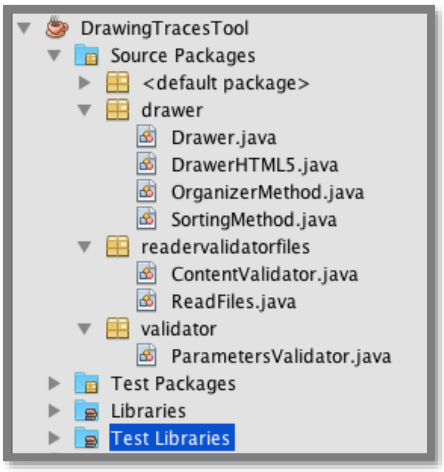


Diagrama de árbol de la herramienta Drawing Traces Tools.

Figura 32:

La Clase DrawerHTML5 se encarga de generar el archivo de salida HTML con el código JavaScript necesario para graficar los elementos que participan en la regulación transcripcional de E. Coli k-12, antes de llegar a esta clase los parámetros de entrada del archivo ya han sido validados y en caso de contener errores aparecerán en la consola del sistema que lo está ejecutando.

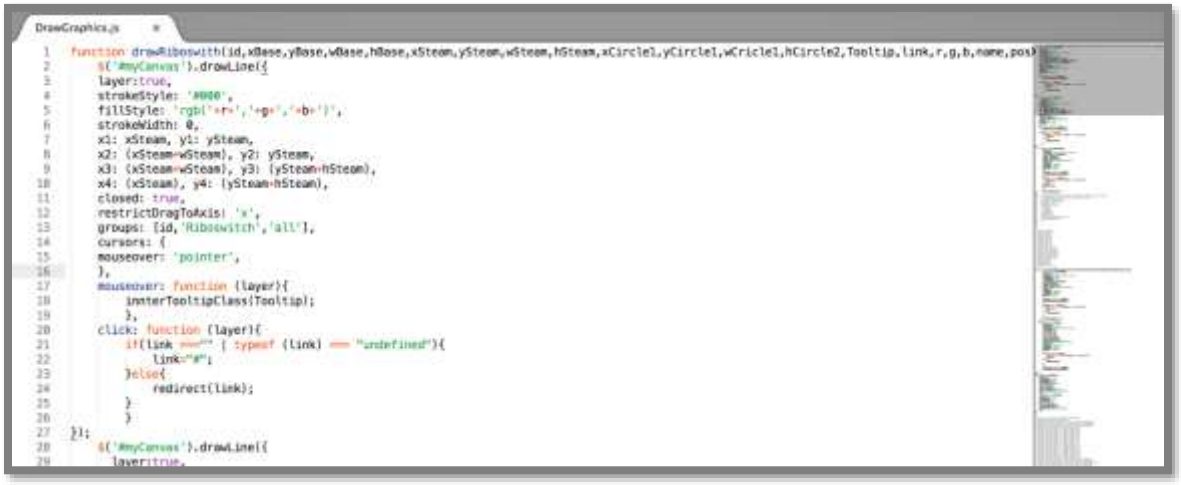
5.6 Desarrollo del API DrawGraphics en JavaScript

Para poder dibujar las imágenes al momento de abrir el archivo html5 en el navegador se desarrolló el API DrawGraphics en el lenguaje JavaScript que será capaz de interpretar el código incrustado en el archivo y generar los elementos

genéticos de manera instantánea. Por cada método genera la función de un elemento en el API RegulonDBGraphics que calcula la posición, colores, borde, tamaño, nombre, información de cada uno se necesita una función en JavaScript.

Esta API fue desarrollada usando otra librería, jCanvas.js por Caleb Evans, para un manejo mejor del lienzo en HTML.

Una ventaja de crear un API en JavaScript es que se puede actualizar la forma de algún elemento genético, cambiar el estándar de colores, eliminar o agregar nuevos elementos de manera inmediata en el servidor sin modificar el código desarrollado en Java.



Vista previa de DrawGraphics.js

5.7 Funcionalidades para la canvas

Una vez teniendo la imagen visualizada en el navegador podemos incluir otras funcionalidades para darle al usuario una mejor experiencia en el sitio, para esto se agregó en un nuevo archivo JavaScript (controllercanvas.js) que funcionara como el controlador del elemento canvas. Este archivo es el encargado de las siguientes funcionalidades (ver imagen 34):

1. Help: Esta función abre una nueva pestaña con el menú de ayuda, describe todas las figuras, sus propiedades y el código de color que pueden utilizar. Este menú de ayuda fue reconstruido.
2. Screen: Toma una captura de pantalla y la muestra al usuario en una nueva pestaña, puede ver la calidad de la imagen antes de descargarla.
3. Save: Descarga directamente la imagen que se está visualizando.
4. Restore: Restaura a su forma original la imagen, en caso de haber utilizado demasiado zoom.
5. Zoom in: Aumenta el tamaño de la imagen a una escala de 1.1 cada clic sin perder calidad en la imagen.
6. Zoom out: Disminuye el tamaño de la imagen a una escala de 0.1 cada vez que es llamada la función, de la misma manera la imagen no pierde calidad.
7. Crop: Sombrea el lienzo y aparece un recuadro expandible para ajustar al tamaño y cortar o extraer un pedazo de la imagen, este fragmento de imagen se abrirá en una nueva pestaña del navegador donde el usuario podrá descargar la imagen si lo desea.

Muestra de los botones

Figura 34:

5.8 Integración de la herramienta DTT a RegulonDBWebApp

Al finalizar la construcción del API en línea de comandos llega la hora de la integración a RegulonDBWebApp. Como se muestra en la Figura 16 del Capítulo 3 - Figura 16: Esquema general del proyecto, RegulonDBWebApp tuvo ligeros cambios para utilizar esta nueva API, el cual fue crear un WebService que regresara la región de la parte del genoma de E. coli k-12 que se está consultando; con esto RegulonDB ya no tiene que generar las 9000 imágenes cada reléase, simplemente un archivo html que dibujara las imágenes a la hora de ser consultadas .

El servlet de RegulonDBWebApp también fue modificado dado que ahora es un ligero cambio, ya no se importa una imagen si no un frame con el archivo html que incluye las funciones para graficar las imágenes, con esto los usuarios pueden descargar sus imágenes a el tamaño que decidan.

Al tener una nueva interface de visualización de genes en RegulonDB se integró el manual de ayuda y código de colores que fue desarrollado en lenguaje markdown y explica todas las figuras que grafica la herramienta DTT asi como los colores que puede utilizar cada una.

CAPITULO 6

PRUEBAS Y RESULTADOS

6.1 Pruebas de Sistema

Uno de los objetivos principales de las pruebas establecidas para el sistema es verificar que como se comporta el software para completar los requisitos establecidos. A medida que aumenta la complejidad de los sistemas software y aumenta la demanda de calidad, se hacen necesarios procesos y métodos que permitan obtener buenos conjuntos de pruebas del sistema. Este capítulo describe las pruebas necesarias para verificar la implementación de los requisitos.

Las pruebas que se realizaron fueron expresamente solicitadas por un grupo de informáticos de Genómica Computacional del CCG, las cuales serán enlistadas de la siguiente manera:

1. Comprobar la generación de los 9 objetos biológicos.
2. Medición de tiempo de graficado en la página web.

3. Visualizar la imagen en los navegadores web (Firefox, Chrome, Safari, Opera, Explorer EDGE).

Se comprobará la ejecución correcta de cada uno de los elementos biológicos que participan en la regulación genética, todos estos serán dibujados individualmente y todos los elementos en una sola imagen.

6.2 Dibujado de los elementos Biológicos

A continuación, se mostrarán los 11 casos de elementos a dibujar, cada uno cuenta con su propio archivo de entrada y su respectivo archivo de salida.

6.2.1 Prueba Gene

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “gene”.

Archivo requerido:

- Gene.txt (Apéndice A)

Resultados esperados:

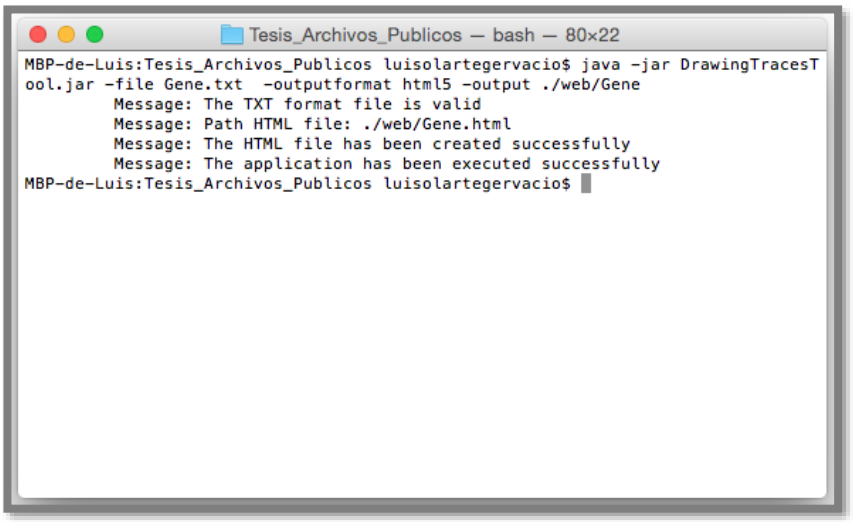
- Ejecución exitosa al ejecutar la herramienta.
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**`java -jar DrawingTracesTool.jar -file Gene.txt -outputformat html5 -output ./web/Gene`**".
- Abrir el archivo de salida llamado Gene.html.

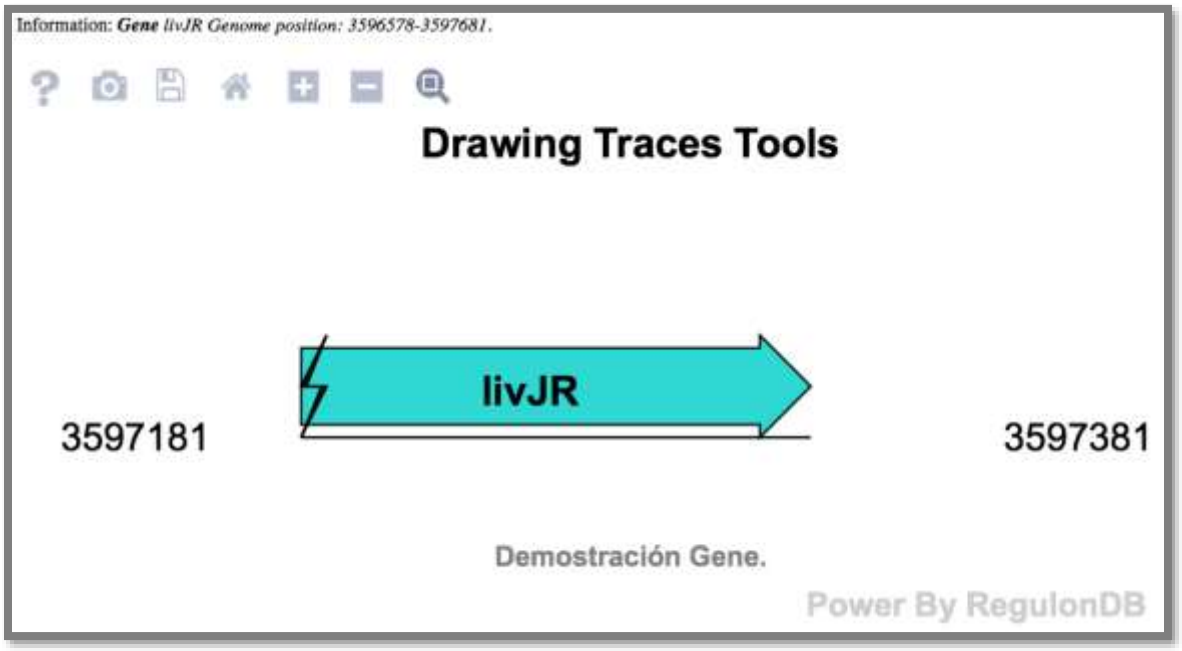
Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo Gene.txt

- Visualización Web



Resultado exitoso Gene

Figura 36:

6.2.2 Prueba Operón

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “operon”.

Archivo requerido:

- operon.txt (Apéndice B)

Resultados esperados:

- Ejecución exitosa al ejecutar la herramienta.

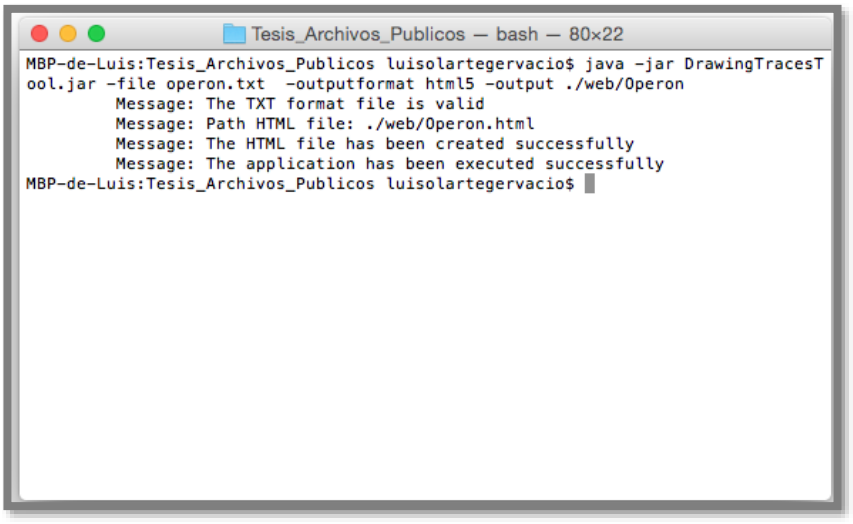
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**`java -jar DrawingTracesTool.jar -file operon.txt -outputformat html5 -output ./web/Gene`**".
- Abrir el archivo de salida llamado Operon.html.

Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo operon.txt

- Visualización Web

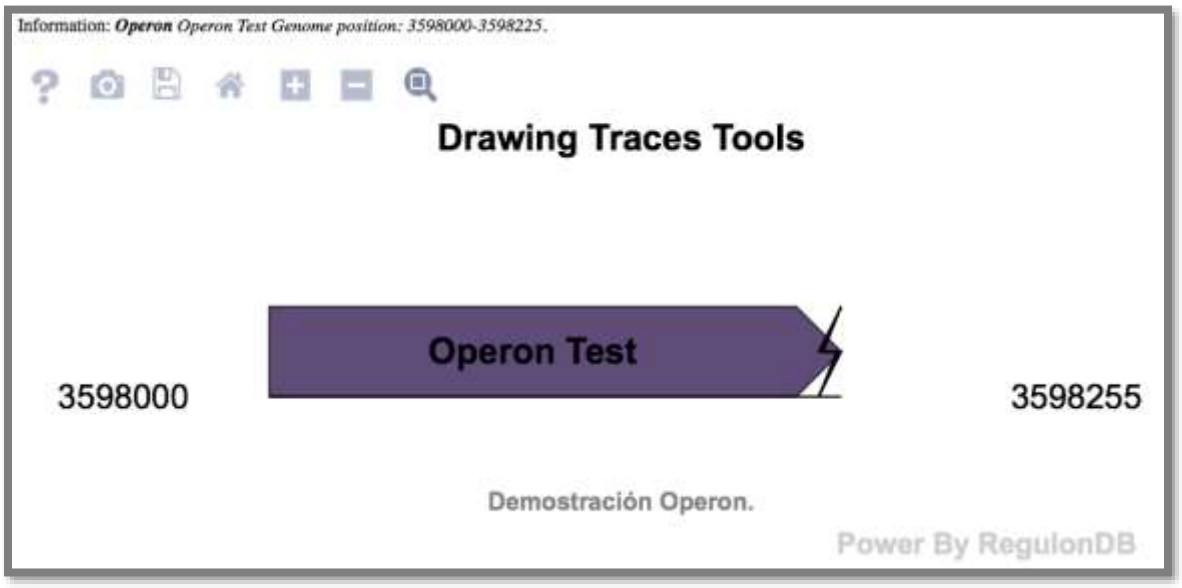


Figura 38: *Resultado exitoso Operón*

6.2.3 Prueba PPGPP

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “ppgpp”.

Archivo requerido:

- ppgpp.txt (Apéndice C)

Resultados esperados:

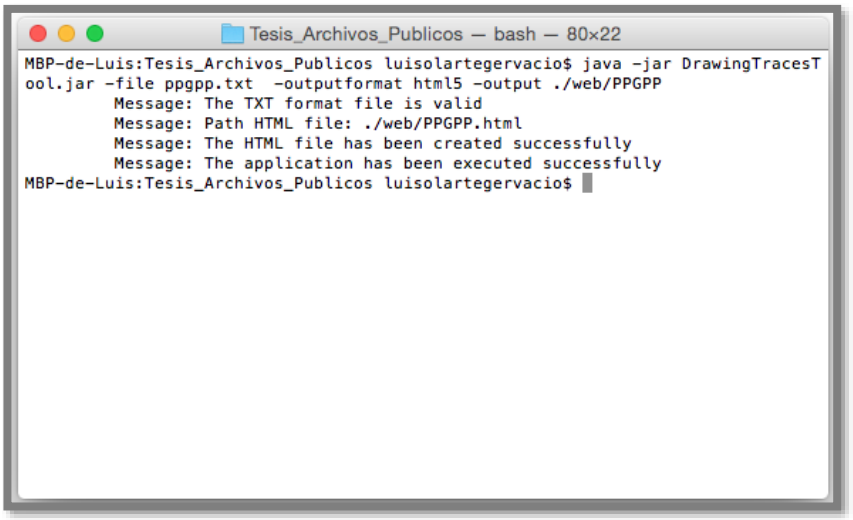
- Ejecución exitosa al ejecutar la herramienta.
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**java -jar DrawingTracesTool.jar -file ppgpp.txt -outputformat html5 -output ./web/PPGPP**".
- Abrir el archivo de salida llamado PPGPP.html.

Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo ppgpp.txt

Figura 39:

- Visualización Web

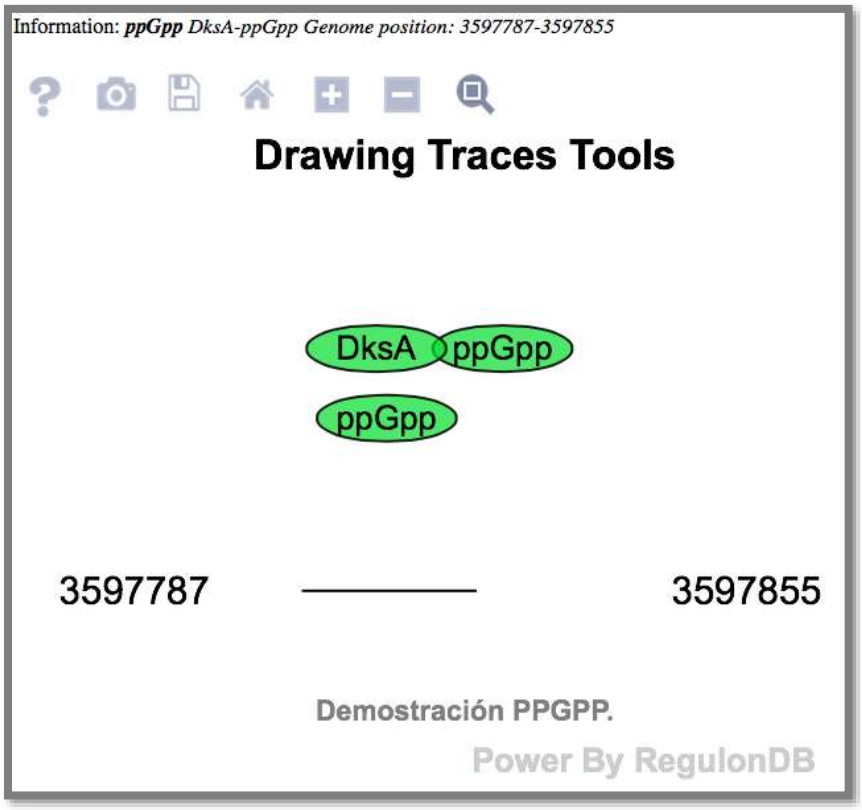


Figura 40: *Resultado exitoso ppgpp*

6.2.4 Prueba Promotor

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “promotor”.

Archivo requerido:

- promotor.txt (Apéndice D)

Resultados esperados:

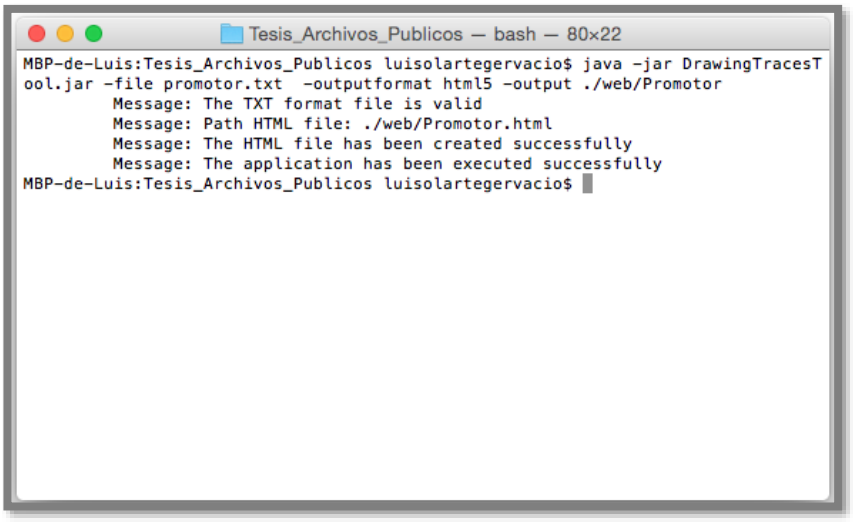
- Ejecución exitosa al ejecutar la herramienta.
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**`java -jar DrawingTracesTool.jar -file promotor.txt -outputformat html5 -output ./web/Promotor`**".
- Abrir el archivo de salida llamado Promotor.html.

Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo promotor.txt

Figura 41:

- Visualización Web

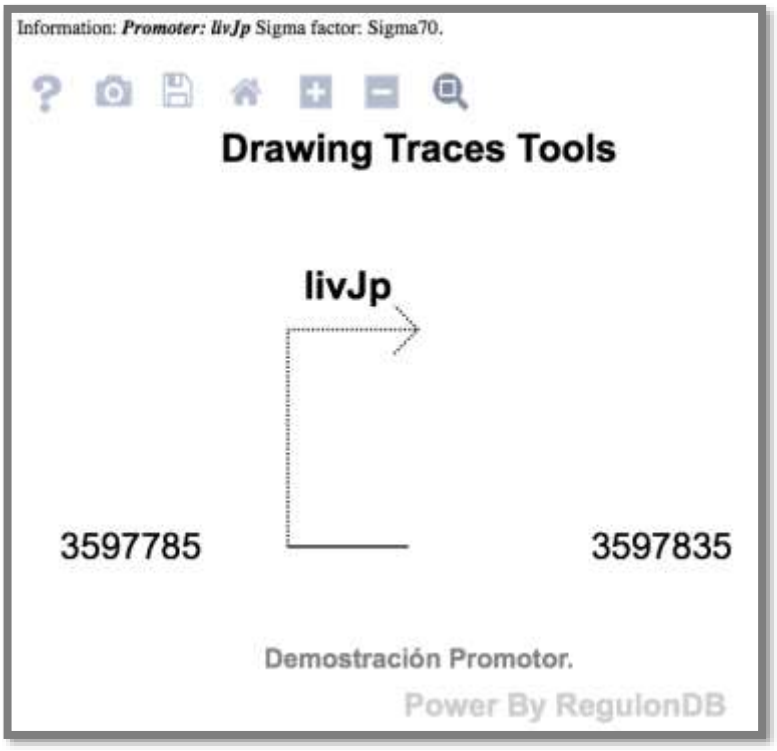


Figura 42: *Resultado exitoso Promotor*

6.2.5 Prueba Riboswitch

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “riboswitch”.

Archivo requerido:

- riboswitch.txt (Apéndice E)

Resultados esperados:

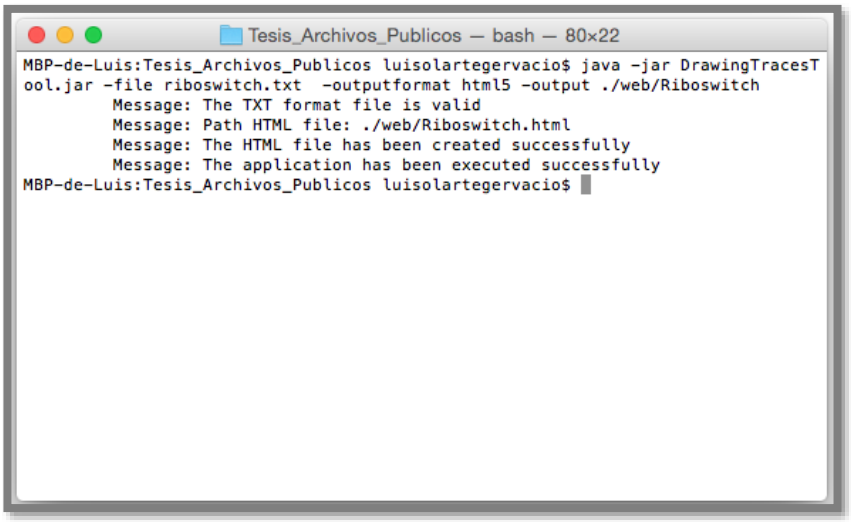
- Ejecución exitosa al ejecutar la herramienta.
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**`java -jar DrawingTracesTool.jar -file riboswitch.txt -outputformat html5 -output ./web/Riboswitch`**".
- Abrir el archivo de salida llamado Riboswitch.html.

Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo riboswitch.txt

Figura 43:

- Visualización Web

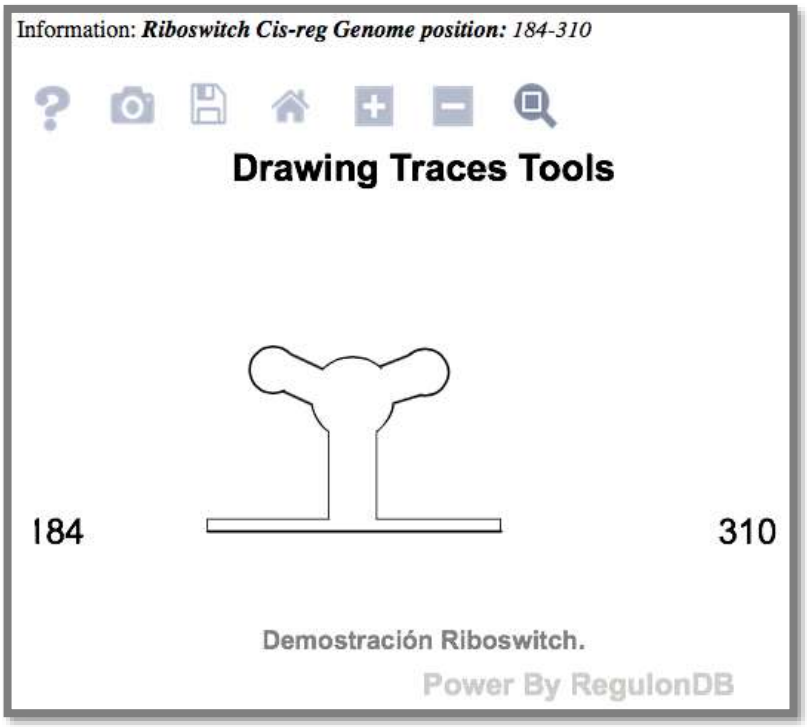


Figura 44: *Resultado exitoso Riboswitch*

6.2.6 Prueba Sitio Unión

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “Sitio Unión”.

Archivo requerido:

- Sitio_union.txt (Apéndice F)

Resultados esperados:

- Ejecución exitosa al ejecutar la herramienta.

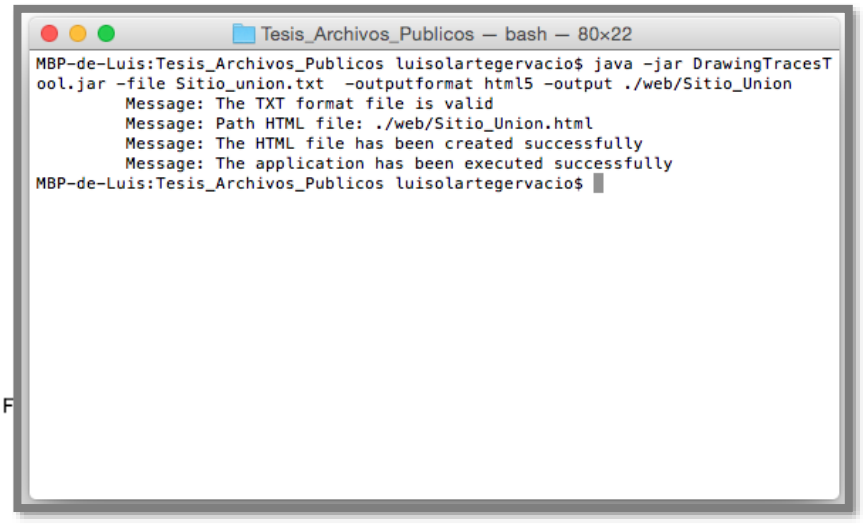
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**`java -jar DrawingTracesTool.jar -file Sitio_union.txt -outputformat html5 -output ./web/Sitio_Union`**".
- Abrir el archivo de salida llamado Sitio_Union.html.

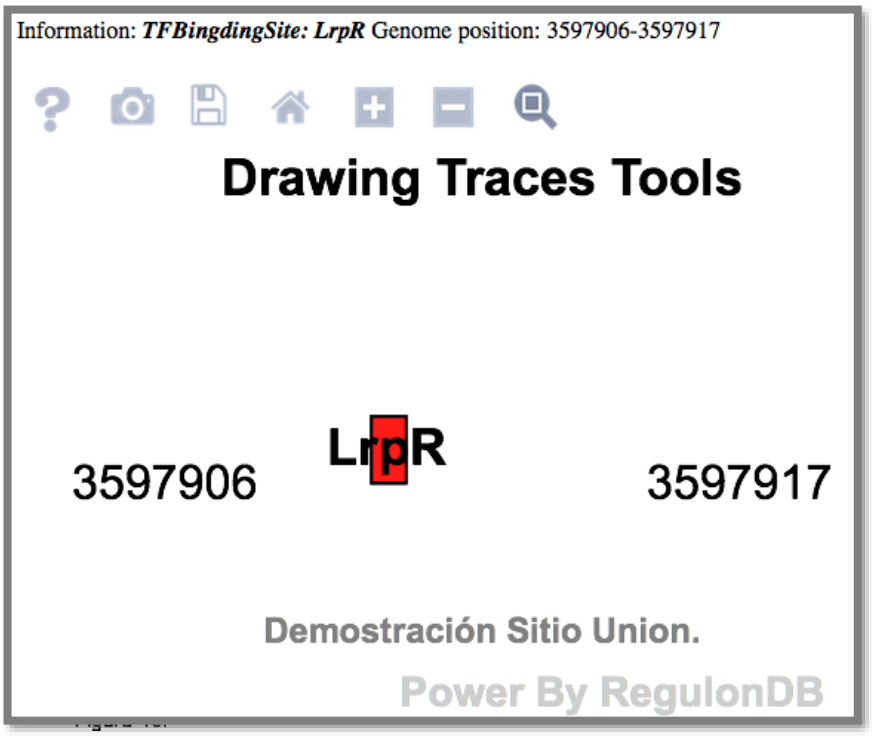
Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo Sitio_union.txt

- Visualización Web



Resultado exitoso Sitio Union

6.2.7 Prueba Srna

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “Srna”.

Archivo requerido:

- srna.txt (Apéndice G)

Resultados esperados:

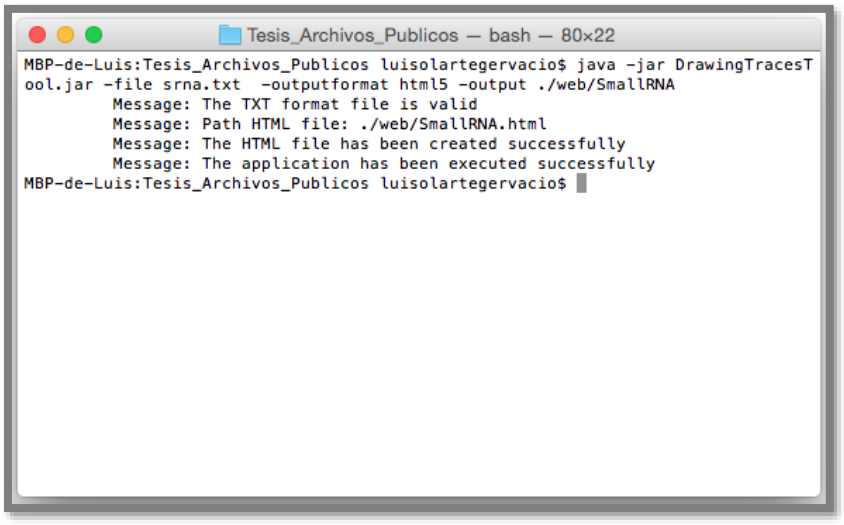
- Ejecución exitosa al ejecutar la herramienta.
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**`java -jar DrawingTracesTool.jar -file srna.txt -outputformat html5 -output ./web/SmallRNA`**".
- Abrir el archivo de salida llamado SmallRNA.html.

Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo srna.txt

Figura 47:

- Visualización Web

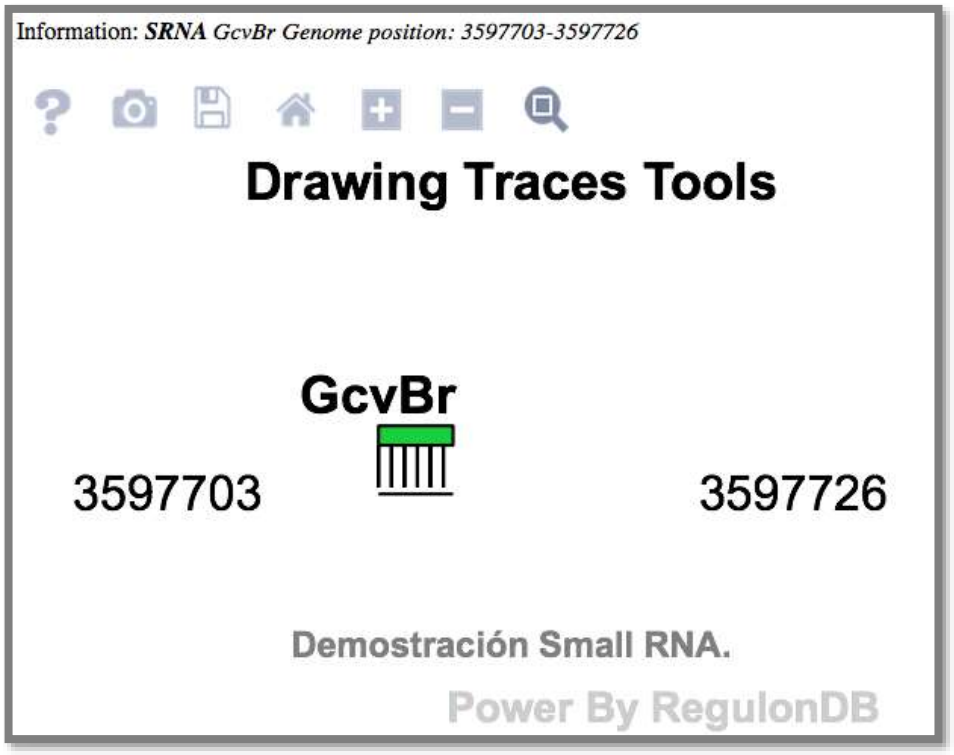


Figura 48: *Resultado exitoso Srna*

6.2.7 Prueba Terminador

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “Terminator”.

Archivo requerido:

- terminador.txt (Apéndice H)

Resultados esperados:

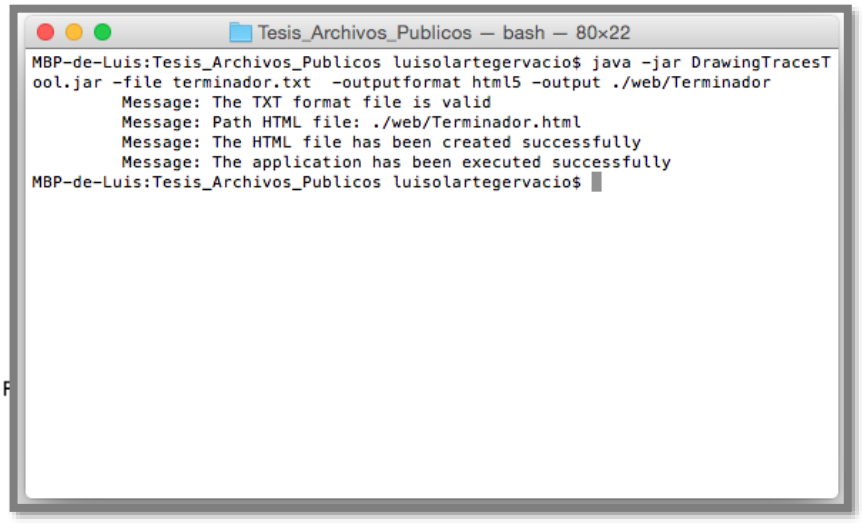
- Ejecución exitosa al ejecutar la herramienta.
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**`java -jar DrawingTracesTool.jar -file terminador.txt -outputformat html5 -output ./web/Terminador`**".
- Abrir el archivo de salida llamado SmallRNA.html.

Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo terminador.txt

- Visualización Web

Resultado exitoso Srna

6.2.8 Prueba Atenuador Transcripcional

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “Transcriptional Attenuator”.

Archivo requerido:

- Transcriptional_attenuator.txt (Apéndice I)

Resultados esperados:

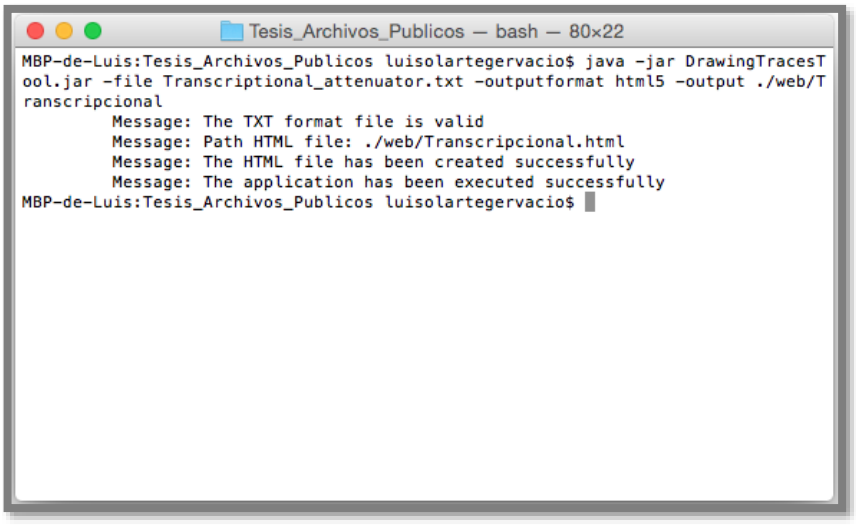
- Ejecución exitosa al ejecutar la herramienta.
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**`java -jar DrawingTracesTool.jar -file Transcriptional_attenuator.txt -outputformat html5 -output ./web/Transcriptional`**".
- Abrir el archivo de salida llamado Transcripcional.html.

Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo terminador.txt

Figura 51:

- Visualización Web

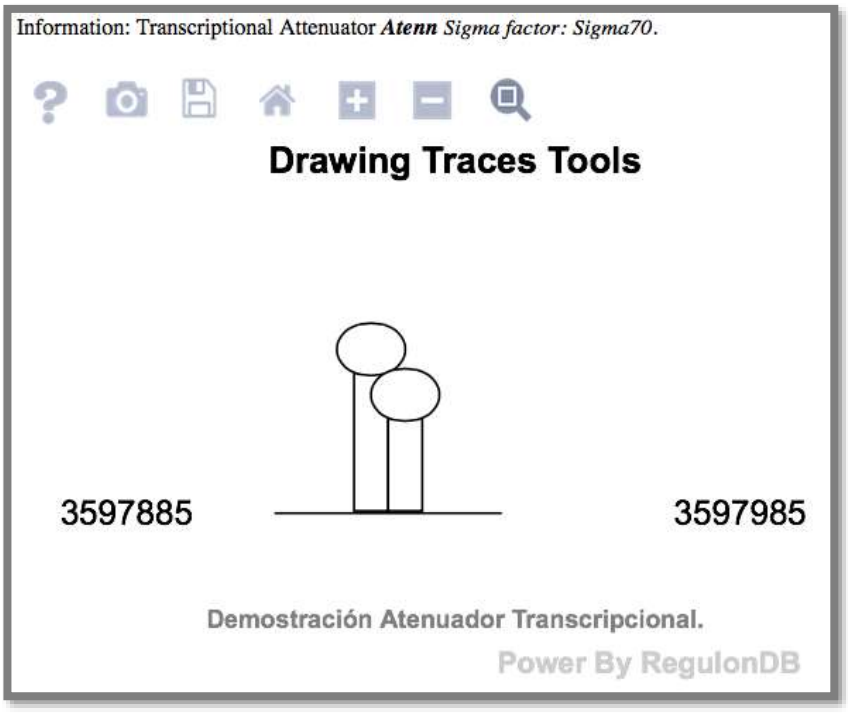


Figura 52: *Resultado exitoso Atenuador Transcripcional*

6.2.9 Prueba Atenuador Traslacional

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado del elemento genético “Translational Attenuator”.

Archivo requerido:

- transcriptional_attenuator.txt (Apéndice J)

Resultados esperados:

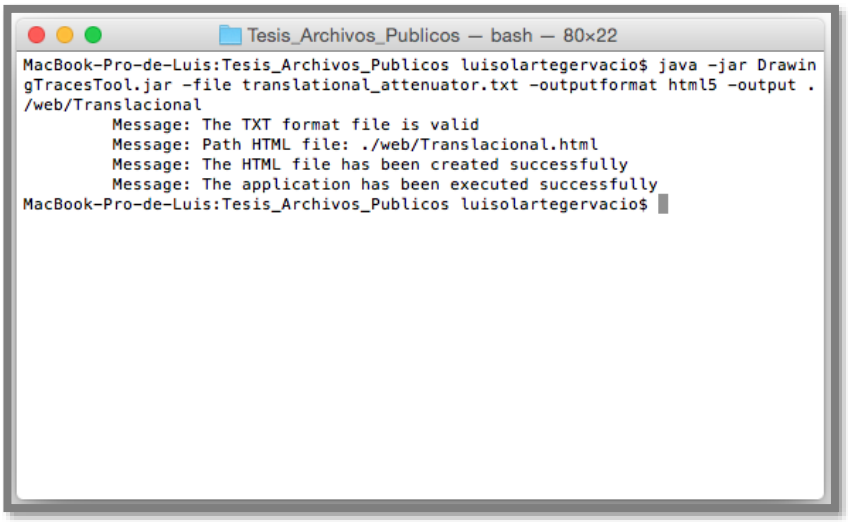
- Ejecución exitosa al ejecutar la herramienta.
- Visualización correcta de la pagina WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**java -jar DrawingTracesTool.jar -file translational_attenuator.txt -outputformat html5 -output ./web/Translational**".
- Abrir el archivo de salida llamado Translational.html.

Resultados obtenidos:

- Salida de terminal:



Salida de Terminal del archivo translational_attenuator.txt

Figura 53:

- Visualización Web

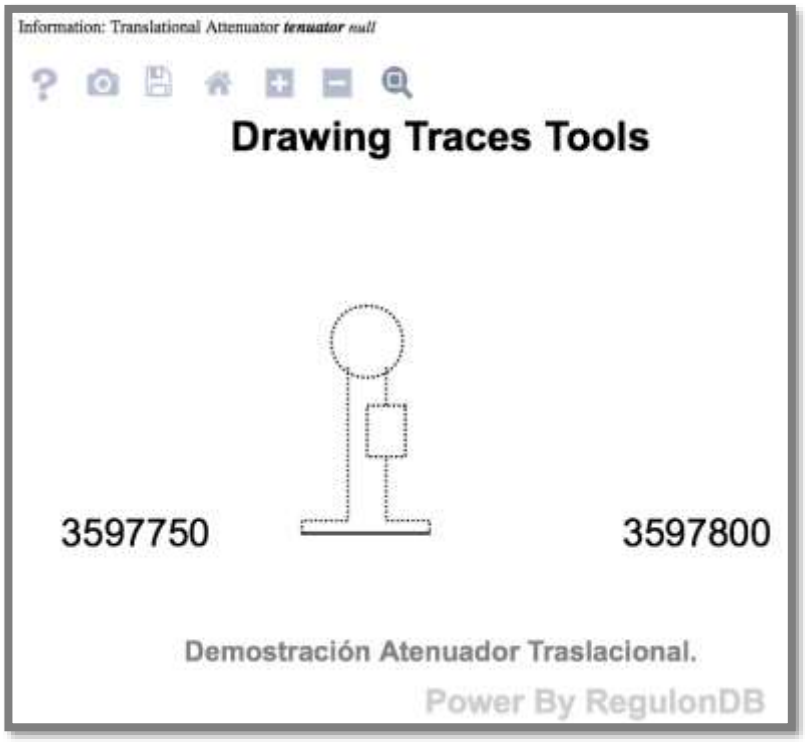


Figura 54: *Resultado exitoso Atenuador Traslacional*

6.2.10 Prueba Todos los Elementos Genéticos

Objetivo de la prueba: Comprobar la ejecución correcta del API DrawingTracesTools.jar y el dibujado de todos los elementos genéticos que participan en la regulación.

Archivo requerido:

- Todos_elementos.txt (Apéndice K)

Resultados esperados:

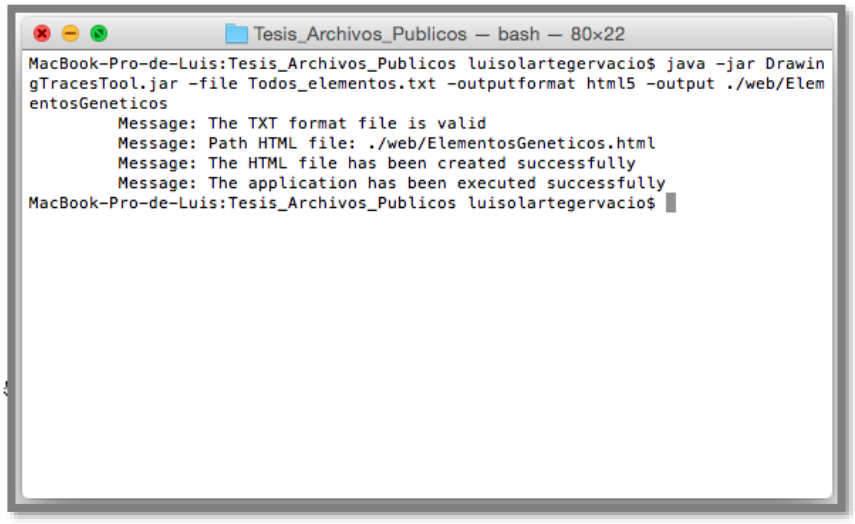
- Ejecución exitosa al ejecutar la herramienta.
- Visualización correcta de la página WEB.

Pasos para realizar la Prueba:

- Desde línea de comandos ejecutar el comando "**`java -jar DrawingTracesTool.jar -file Todos_elementos.txt -outputformat html5 -output ./web/ElementosGeneticos`**".
- Abrir el archivo de salida llamado Translational.html.

Resultados obtenidos:

- Salida de terminal:



Figura

Salida de Terminal del archivo translational_attenuator.txt

- Visualización Web

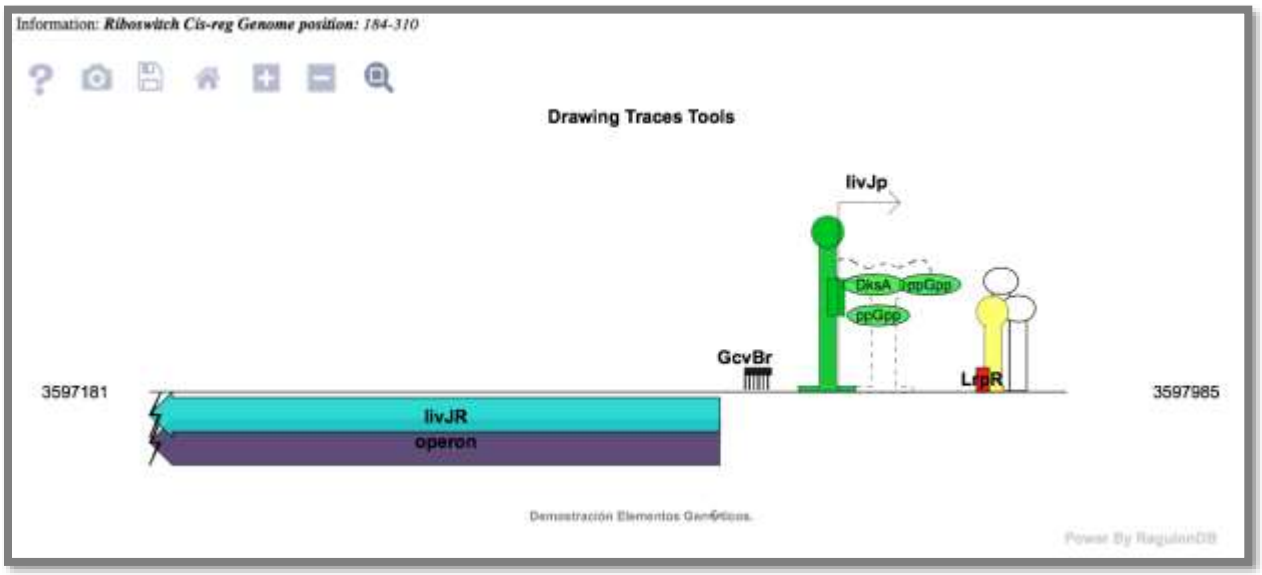


Figura 56: *Resultado exitoso Elementos Genéticos.*

CAPITULO 7:

CONCLUSIONES Y TRABAJOS FUTUROS

7.1 Conclusiones

En el presente trabajo de tesis se desarrolló un API para el graficado de elementos genéticos que participan en la regulación genética la cual permite la visualización de la información almacenada en la base de datos RegulonDB utilizada por investigadores y curadores del Programa de Genómica Computacional de la UNAM, el API fue probada y validada por un grupo de pruebas arrojado resultados satisfactorios.

El API Drawing Traces Tools cuenta con la capacidad de obtener datos vaciados en archivos tipo json o txt, con información de la base de datos RegulonDB para poder ser procesados y dibujados en visualizados en cualquier navegador web. También tiene otras capacidades como aumentar o disminuir el tamaño de la imagen proporcionalmente evitando así, perdida de calidad al descargar la imagen; es posible seleccionar una parte del genoma y extraerla, es decir en caso de no querer descargar la imagen el usuario obtiene la fracción de

imagen que desea y extraerla en una nueva ventana web donde podrá descargarla o tomar una nueva fracción de imagen.

El API es capaz de crear el nuevo archivo html generado y este podrá ser portado a cualquier parte que el usuario desee.

El servidor de RegulonDB ahora pude liberar espacio eliminando las imágenes previamente procesadas con lo cual al implementar esta nueva herramienta los usuarios podrán generar una imagen con más calidad sin la necesidad que sea generada y almacenada en el servidor, actualmente RegulonDB genera una nueva imagen si no existe en el servidor y es mostrada al usuario.

Todas las consultas que se hacen actualmente en la página web gene de RegulonDB muestran el resultado del API que genera la imagen al momento de consultarla a través de JavaScript lo que hace que la velocidad de respuesta sea inmediata y trabajado del lado del usuario, sin embargo, si los usuarios no permiten la ejecución de JavaScript en sus navegadores web la imagen no se visualizara.

Cada parte del API está diseñado y modelado con UML siguiendo los lineamientos que establece la ingeniería de software teniendo todos los documentos necesarios para seguir trabajando en actualizaciones futuras.

De acuerdo con todo lo anteriormente expresado se considera que se cumplieron los objetivos establecidos en el trabajo de forma satisfactoria.

7.2 Trabajo a Futuros.

En este proyecto se puede continuar si se realiza para la implementación de otras bacterias dado que el diseño de los elementos genéticos es similar. La implementación de este desafío no requerirá de un modelo nuevo para soportar toda la nueva información ya que el sistema está desarrollado en base a módulos.

Otras funcionalidades extras que se pueden implementar en es desaparecer tipos de elementos genéticos, en caso de que el usuario solo desee ver tipos de elementos o desparecer algún otro. Implementar esto es una manera mas fácil ya que todo es en lenguaje JavaScript y los cambios son instantáneos.

Apéndice A: Gene.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Gene.
%object_type	left_position	right_position	strand	object_color	line_width	line_type
	label_name	label_font	label_color	label_size	tooltip	line_color	href
gene	+3597181	3597381	forward	0,209,204	1	0	livJR	arial	0,0,0	12	Genome position:<br> 3596578-3597681.	0,0,0
	http://regulondb.ccg.unam.mx/gene?organism=ECK12&term=ECK120000574&format=jsp&type=gene
```

Apéndice B: operon.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Operon.
%object_type	left_position	right_position	strand	object_color	line_width	line_type
	label_name	label_font	label_color	label_size	tooltip	line_color	href
operon	3598000	3598255+	forward	25,0,66	1	0	Operon Test	arial	0,0,0	12
	Genome position:<br> 3598000-3598225.	0,0,0
```

Apéndice C: ppgpp.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& PPGPP.
%object_type	left_position	right_position	strand	object_color	line_width	line_type
	label_name	label_font	label_color	label_size	tooltip	line_color	href
ppGpp	3597787	3597820	forward	19,221,0	1	1	ppGpp	arial	0,0,0	10	Genome position: 3597787-3597820	0,0,0
ppGpp	3597787	3597855	forward	19,221,0	1	1	DksA-ppGpp	arial	0,0,0	10
	Genome position: 3597787-3597855	0,0,0
```

Apéndice D: promotor.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Promotor.
%object_type	left_position	right_position	strand	object_color	line_width	line_type
	label_name	label_font	label_color	label_size	tooltip	line_color	href
promoter	3597785	3597785	forward	0,0,0	1	1	livJp	arial	0,0,0	12	Sigma factor: <br> Sigma70.	0,0,0
```

Apéndice E: riboswitch.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Riboswitch.
%object_type	left_position	right_position	strand	object_color	line_width	line_type
	label_name	label_font	label_color	label_size	tooltip	line_color	href
riboswitch	184	310	forward	255,255,255	1	0	Cis-reg	arial	0,0,0	12
	<strong>Genome position:</strong><br> 184-310	0,0,0
```

Apéndice F: Sitio_Union.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Sitio Union.
%object_type	left_position	right_position	strand	object_color	line_width	line_type
	label_name	label_font	label_color	label_size	tooltip	line_color	href
tf_binding_site	3597906	3597917	forward	255,28,0	1	0	LrpR	arial	0,0,0	12
	Genome position: 3597906-3597917	0,0,0
```

Apéndice G: srna.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Small RNA.
%object_type	left_position	right_position	strand	object_color	line_width	line_type
	label_name	label_font	label_color	label_size	tooltip	line_color	href
srna	3597703	3597726	forward	0,200,0	1	0	GcvBr	arialR	0,0,0	12	Genome
position: 3597703-3597726	0,0,0
```

Apéndice H: terminator.txr

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Terminador.
%object_type	left_position	right_position	strand	object_color	line_width	line_type
	label_name	label_font	label_color	label_size	tooltip	line_color	href
terminator	3597910	3597930	forward	255,255,255	1	1	terminator	arial	0,0,0	12
	rho-independent.	0,0,0
```

Apéndice I: Transcriptional_attenuator.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Atenuador Transcripcional.
%object_type	left_position	right_position	strand	object_color	line_width	line_type
	label_name	label_font	label_color	label_size	tooltip	line_color	href
```

Transcriptional_attenuator		3597885	3597985	forward	255,255,255		1	0	Atenn	arial
0,0,0	12	Sigma factor: Sigma70.			0,0,0					

Apéndice J: translational_attenuator.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Atenuador Traslacional.
%object_type left_position right_position strand object_color line_width line_type
label_name label_font label_colorlabel_size tooltip line_color href
translational_attenuator 3597750 3597800 forward 255,255,255 1 1 tenuator
```

Apéndice K: Todos_elementos.txt

```
# Generated file with database RegulonDB
* Drawing Traces Tools
& Elementos Genéticos.
%object_type left_position right_position strand object_color line_width line_type
label_name label_font label_colorlabel_size tooltip line_color href
gene +3597181 3597681 reverse 0,209,204 1 0 livJR arial 0,0,0 12 Genome
position:<br> 3596578-3597681. 0,0,0 ?organism=ECK12&term=ECK120000574&format=jsp&type=gene
operon +3597181 3597681 reverse 25,0,66 1 0 operon arial 0,0,0 12 Genome
position:<br> 3598000-3598225. 0,0,0
ppGpp 3597787 3597820 forward 19,221,0 1 0 ppGpp arial 0,0,0 10 Genome
position: 3597787-3597820 0,0,0
ppGpp 3597787 3597855 forward 19,221,0 1 0 DksA-ppGpp arial 0,0,0 10
Genome position: 3597787-3597855 0,0,0
promoter 3597785 3597785 forward 0,0,0 1 1 livJp arial 0,0,0 12 Sigma
factor: <br> Sigma70. 0,0,0
tf_binding_site 3597906 3597917 forward 255,28,0 1 0 LrpR arial 0,0,0 12
Genome position: 3597906-3597917 0,0,0
srna 3597703 3597726 forward 0,0,0 1 0 GcvBr arialR 0,0,0 12 Genome
position: 3597703-3597726 0,0,0
terminator 3597910 3597930 forward 255,255,255 1 1 terminator arial 0,0,0 12
rho-independent. 0,0,0
Transcriptional_attenuator 3597885 3597985 forward 255,255,255 1 0 Atenn arial
0,0,0 12 Sigma factor: <br> Sigma70. 0,0,0
translational_attenuator 3597750 3597800 forward 0,190,0 1 1 tenuator
riboswitch 3597800 3597850 forward 255,255,255 1 5 Cis-reg arial 0,0,0 12
<strong>Genome position:</strong><br> 184-310 0,0,0
```

Referencias

1. Alonso, A., *Una propuesta de lenguaje exclusivamente gráfico para la comunicación humana.* Comunicación, Lenguaje y Educación, 1992. **4**(14): p. 113-124.

2. Wikipedia, c.d. *Visualización de datos*. Available from: https://es.wikipedia.org/w/index.php?title=Visualizaci%C3%B3n_de_datos&oldid=102849788.

3. Gama-Castro, S., et al., *RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond.* Nucleic Acids Res, 2016. **44**(D1): p. D133-43.

4. Salgado, H., et al., *RegulonDB v8.0: omics data sets, evolutionary conservation, regulatory phrases, cross-validated gold standards and more.* Nucleic Acids Res, 2013. **41**(Database issue): p. D203-13.

5. Gama-Castro, S., et al., *RegulonDB version 7.0: transcriptional regulation of Escherichia coli K-12 integrated within genetic sensory response units (Gensor Units).* Nucleic Acids Res, 2011. **39**(Database issue): p. D98-105.

6. Gama-Castro, S., et al., *RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation.* Nucleic Acids Res, 2008. **36**(Database issue): p. D120-4.

7. Salgado, H., et al., *RegulonDB (version 5.0): Escherichia coli K-12 transcriptional regulatory network, operon organization, and growth conditions.* Nucleic Acids Res, 2006. **34**(Database issue): p. D394-7.

8. Salgado, H., et al., *RegulonDB (version 4.0): transcriptional regulation, operon organization and growth conditions in Escherichia coli K-12.* Nucleic Acids Res, 2004. **32**(Database issue): p. D303-6.

9. Salgado, H., et al., *RegulonDB (version 3.2): transcriptional regulation and operon organization in Escherichia coli K-12.* Nucleic Acids Res, 2001. **29**(1): p. 72-4.

10. Salgado, H., et al., *RegulonDB (version 3.0): transcriptional regulation and operon organization in Escherichia coli K-12.* Nucleic Acids Res, 2000. **28**(1): p. 65-7.

11. Salgado, H., et al., *RegulonDB (version 2.0): a database on transcriptional regulation in Escherichia coli.* Nucleic Acids Res, 1999. **27**(1): p. 59-60.

12. Huerta, A.M., et al., *RegulonDB: a database on transcriptional regulation in Escherichia coli.* Nucleic Acids Res, 1998. **26**(1): p. 55-9.

13. Palacio, J.D.C., *Diseño e implementación de un sistema en web de biblioteca digital de documentos de literatura científica*. 2005, Universidad Tecnológica de la Mixteca: Mexico. p. 179.

14. Wikipedia, c.d. *Lenguaje de programación* 2017 Agosto, 2017]; Available from: https://es.wikipedia.org/w/index.php?title=Lenguaje_de_programaci%C3%B3n&oldid=101008232.

15. Gauchat, J.D., *El gran libro de HTML5, CSS3 y JavaScript: Marcombo ediciones técnicas.* Obtenido de: http://www/. cunlimon. ac. cr/Uploads/InfoPublica/HTML5_CSS3_Javascript. pdf, accesado el, 2014: p. 14-08.

16. Wikipedia, c.d. *Hoja de estilos en cascada*. Agosto, 2017]; Available from: https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada.

17. Gruber, J., *Daring Fireball: Markdown.* Récupéré le, 2004. **3**(04): p. 2011.

18. Deitel, H.M. and P.J. Deitel, *Cómo programar en Java*. 2004: Pearson Educación.

19. Larman, C., *UML y Patrones*. 1999: Pearson.

20. González, Y.D. and Y.F. Romero, *Patrón Modelo-Vista-Controlador.* Revista Telem@ tica, 2012. **11**(1): p. 47-57.

21. Tinoco, A.R., *Cartografía automática en Internet.* Equipo de investigación Varilex, 2002: p. 6-17.

22. Rascado, L.J.M., *Desarrollo de una bodega de datos para el almacenamiento de información génica sobre el proceso de transcripción de múltiples bacterias*, in *Sistemas y Computacion*. 2007, Instituto Tecnológico de Zacatepec: Mexico. p. 93.

23. Reference, G.H. *What is a genome?* . NLM 2017 [cited 2017 Febrero, 2017]; Available from: https://ghr.nlm.nih.gov/primer/hgp/genome.

24. Medicine, U.S.N.L.o. *Cortesía de la Biblioteca Nacional de Medicina de los Estados Unidos, Bethesda, Maryland.* 2017.

25. Medicine, U.S.N.L.o. *What is a chromosome?* 2017 Febrero, 2017]; Available from: https://ghr.nlm.nih.gov/primer/basics/chromosome.

26. Raisman, D.J.S. *Regulación genética en Procariotas*. 2013 [cited Agosto, 2017; Available from: http://www.biologia.edu.ar/adn/adntema3.htm.

27. Baba, T., et al., *Construction of Escherichia coli K-12 in-frame, single-gene knockout mutants: the Keio collection.* Molecular Systems Biology, 2006. **2**(1).

28. Wikipedia, c.d. *Regulación génica en procariotas*. Agosto, 2017]; Regulación génica en procariotas]. Available from: https://es.wikipedia.org/w/index.php?title=Regulaci%C3%B3n_g%C3%A9nica_en_procariotas&oldid=99776209.

29. Chhabra, D.N. *Lac operon (Regulation of gene expression in Prokaryotes)*. 2013 Agosto, 2017]; Available from: http://www.namrata.co/lac-operon-regulation-of-gene-expression-in-prokaryotes/.

30. RegulonDB. *RegulonDB Glossary*. 2017 [cited 2017 Febrero, 2017]; Available from: http://regulondb.ccg.unam.mx/menu/using_regulondb/tutorials/project_glossary/index.jsp.

31. Biology, M., *Promoter site*. 2017.

32. Academy, K. *Factores de la transcripción*. 2017 Febrero, 2017]; Available from: https://es.khanacademy.org/science/biology/gene-regulation/gene-regulation-in-eukaryotes/a/eukaryotic-transcription-factors.

33. Wikipedia, c.d. *Riboswitch*. Agosto, 2017]; Available from: https://es.wikipedia.org/w/index.php?title=Riboswitch&oldid=100265885.

34. Claros, M.G. *Riborreguladores o ribointerruptores*. Febrero, 2017]; Available from: http://www.biorom.uma.es/contenido/av_bma/apuntes/T4/riboreg.html.

35. Wikipedia, c. *Trp operon attenuation*. February 2011 [cited Febrero, 2017; Available from: https://commons.wikimedia.org/wiki/File:Trp_operon_attenuation.svg.

36. Yanofsky, C., *The different roles of tryptophan transfer RNA in regulating trp operon expression in E. coli versus B. subtilis.* Trends Genet, 2004. **20**(8): p. 367-74.

37. Swanson, Z.D.D.M.S. *FIGURE 1 | ppGpp alters RNA polymerase promoter selection by σ-factors.* 2012 Febrero, 2017]; Available from: http://www.nature.com/nrmicro/journal/v10/n3/fig_tab/nrmicro2720_F1.html.

38. Norris, M. and P. Rigby, *Ingeniería de software explicada*. 1994.