Universitat de Vic
Escola Politècnica Superior

Treball Final de Carrera

Visualitzador de molècules d'ADN per laboratori Virtual

Ivan López Garzón

Enginyeria Tècnica en Informàtica de Gestió i
Enginyeria Tècnica en Informàtica de Sistemes

Directors: Albert Baucells i Colomer
Agustí Fontarnau i Riera

Vic, juny de 2009

Voldria agrair a tota la gent que ha fet possible aquest projecte directe o indirectament amb la seva ajuda i suport incondicional.

Als directors del projecte Albert Baucells i Agustí Fontarnau per oferir-me el projecte, dirigir-me i aguantar-me amb una gran paciència. Als companys i amics Jordi i Toni que m'han ajudat amb els seus consells i suport. Als companys de feina que amb canvis d'horaris, programari i ànims, m'han ajudat a poder finalitzar el projecte. A Curri per l'ajuda amb la traducció. A tots els amics que em demanaven com anava el projecte i em distreien quan ho necessitava.

A la meva família pel ànims, paciència i comprensió que han tingut tot aquest temps.

Lara, a tu, per tot.

Moltes gràcies a tots.

Índex

Universitat de Vic
Escola Politècnica Superior

Resum de treball final de carrera
Enginyeria Tècnica en Informàtica de Gestió
Enginyeria Tècnica en Informàtica de Sistemes

Titol: Visualitzador de molècules d'ADN per laboratori Virtual
Paraules claus: Visualitzador, molècules, ADN, laboratori, Virtual, php, Flash, Perl, ActionScript
Autor: Ivan López Garzón
Direcció: Albert Baucells i Colomer, Agustí Fontarnau i Riera
Data: juny del 2009

Resum

Com a continuació del treball de final de carrera "Desenvolupament d'un laboratori virtual per a les pràctiques de Biologia Molecular" de Jordi Romero, s'ha realitzat una eina complementaria per a la visualització de molècules integrada en el propi laboratori virtual.

Es tracta d'una eina per a la visualització gràfica de gens, ORF, marques i seqüències de restricció de molècules reals o fictícies. El fet de poder treballar amb molècules fictícies és la gran avantatge respecte a les solucions com GENBANK que només permet treballar amb molècules pròpies. Treballar amb molècules fictícies fa que sigui una solució ideal per a l'ensenyament, ja que dóna la possibilitat als professors de realitzar exercicis o demostracions amb molècules reals o dissenyades expressament per a l'exercici a demostrar.

A més, permet mostrar de forma visual les diferents parts simultàniament o per separat, de manera que ofereix una primera aproximació interpretació dels resultats. Per altra banda, permet marcar gens, crear marques, localitzar seqüències de restricció i generar els ORF de la molècula que nosaltres creem o modificar una ja existent.

Per l'implementació, s'ha continuat amb l'idea de separar la part de codi i la part de disseny en les aplicacions Flash. Per fer-ho, s'ha utilitzat la plataforma de codi lliure Ariware ARPv2.02 que proposa un marc de desenvolupament d'aplicacions Flash orientades a objectes amb el codi (classes ActionScript 2.0) separats del *movieclip*. Per al processament de dades s'ha fet servir Perl per ser altament utilitzat en Bioinformàtica i per velocitat de càlcul. Les dades generades es guarden en una Base de Dades en MYSQL (de lliure distribució), de la que s'extreuen les dades per generar fitxers XML, fent servir tant PHP com la plataforma AMFPHP com a enllaç entre Flash i la resta de parts.

Universitat de Vic
Escola Politècnica Superior

Summary of the Degree essay
Technical Engineering in Computing Management
Technical Engineering in Computer Systems

Title: DNA Molecules display screen for a virtual laboratory
Keywords: Display screen, molecules, DNA, laboratory, virtual, php, Flash, Perl, ActionScript
Author: Ivan López Garzón
Guiding: Albert Baucells i Colomer, Agustí Fontarnau i Riera
Date: June 2009

Summary

Continuing with the Degree essay “Developing a virtual laboratory for the Molecular Biology practises” by Jordi Romero, a complementary tool has been made in order to visualise the integrated molecules at the virtual laboratory.

This tool graphically visualises genes, ORF, restriction marks and sequences in real or fictitious molecules. Being able to work with fictitious molecules is a great advantage respect to solutions like GENBANK, which only allows working with molecules within GENBANK itself. Working with fictitious molecules is the ideal solution in education, as the teachers have the possibility to do exercises or demonstrations with either real molecules or molecules designed specifically for the exercise to show.

It allows to visually show different parts, either simultaneously or separately, therefore offering a first approximation and interpretation of the results. It allows marking genes, creating marks, localising restriction sequences and either generating the ORF for the molecule created by us or modifying an existing one.

When doing the implementation, I separated the code part from the design parts within the Flash applications. In order to do that, I used Ariware ARPv2.02, a free code platform, which proposes the development of Flash applications oriented to object with the code (classes ActionScript 2.0), separated from the movie clip. When processing the data I used Perl, as it is highly used in Biocomputing and fast in calculating. The generated data is saved in a Database in MYSQL (free distribution), from which data is generated to create XML files, using PHP and AMFPHP platform as a link between Flash and the rest of the parts.

1 Objectiu del projecte

L'objectiu del projecte és la realització d'una aplicació que formarà part del Treball Final de Carrera de Jordi Romero i Sallent amb el nom "Desenvolupament d'un laboratori virtual per a les pràctiques de Biologia Molecular". Una aplicació que s'integrarà en el sistema LabVir i que permetrà la visualització, de forma senzilla e intuïtiva, de molècules d'ADN i conceptes relacionats amb el ADN.
Es podran utilitzar molècules ja existents en altres entorns com per exemple GENBANK o bé crear una molècula des de zero. Utilitzar molècules ja existents serà tant senzill com descarregar el fitxer ".fasta" de qualsevol base de dades d'altres entorns i pujar l'arxiu a l'aplicació. Tant si utilitzem una molècula existent com si creem una nova, els haurem d'assignar un nom i un fitxer (.fasta) associat.

Un cop creada la molècula podrem procedir a entrar les diferents parts que volem marcar, ja siguin gens, marques, localitzar restriccions o generar els ORF de la molècula.

Amb la molècula i les diferents parts entrades podrem visualitzar-les per separat o conjuntament; oferint la possibilitat de mostrar diferents capes a la vegada (Gens, marques, ORF, seqüències), modificar les visualitzacions o modificar el propi codi de la seqüència d'ADN, per així crear molècules totalment diferents de l'original.

L'eina permetrà fer zoom de la molècula que s'està tractant per una millor lectura visual de les diferents parts mostrades.

L'aplicació permetrà a l'usuari descarregar el fitxer (.fasta) associat a la molècula que està visualitzant.

Si bé existeixen eines molt més professionals que la realitzada, aquesta ofereix l'avantatge de poder treballar amb noves molècules encara no conegudes o simplement inventades, donant una primera aproximació visual de les parts que vulguem remarcar conforme es vagin trobant, com per exemple els gens o localització de seqüències. Treballa indistintament amb molècules normals o circulars, calculant els ORF automàticament.

2 Introducció a la Biologia Molecular

2.1 Conceptes essencials

L'aplicació treballa amb molècules d'ADN.

L'**ADN** o **DNA** és el material genètic del éssers vius. És el component químic primari dels cromosomes i el material del que els gens estan formats. Conté informació hereditària on es codifica la síntesis de proteïnes, lípids i altres constituents de l'organisme. Les molècules d'ADN són polímers de quatre monòmers anomenats **desoxirribonucleòtids** o **nucleòtids**. Els nucleòtids tenen tres parts, la part que distingeix els uns dels altres s'anomena "base nitrogenada" o "base". A la pràctica es fan servir només els noms de les bases per representar els nucleòtids (les inicials):

A, de Adenina
G, de Guanina
C, de Citosina
T, de Timina

Per tant una molècula d'ADN tindria un aspecte semblant a:

GATGTAGAAATGGTGAGT...

o en minúscules

gatgtagaaatggtgagt...

Les molècules d'ADN poden ser de només una cadena, anomenades de **cadena simple** o de dues cadenes, anomenades **cadena doble**. En cadenes dobles l'estabilitat de la molècula ve donada per enllaços febles, entre parells de bases A – T i G – C. Els diferents parell de bases s'uneixen lateralment per enllaços forts, formant dues cadenes que es retorcen l'una al voltant de l'altre creant una hèlix.

Cadena simple (ssDNA): GATTACACAGATATGACC...

Cadena doble (dsDNA): GATTACACAGATATGACC...
CTAATGTGTCTATACTGG...

Donada la complementarietat entre les bases, les cadenes dobles també es representen generalment com a cadenes simples, indicant que es tracta d'ADN de cadena doble.

Les molècules d'ADN poden ser de centenars de milions de bases o parells de bases, en una única seqüència. El nombre total de bases d'una molècula és una manera d'indicar la seva grandària que s'expressa en pb (parells de bases), kpb (milers de parells de bases) o Mpb (milions de parells de bases).

2.2 Orientació de les molècules

L'ADN, així com altres classes de molècules, estan polaritzades, és a dir, els seus dos extrems són químicament diferents. Aquesta polarització es manifesta en la seva correcta orientació a l'hora d'escriure-les.

Els extrems de les molècules d'ADN, s'indiquen amb 5' i 3' i es representen amb l'extrem 5' a l'esquerra i 3' a la dreta. Un exemple seria:

ADN de simple cadena (ssDNA):

5'- GATTACACAGATATGACC... – 3'

ADN de doble cadena (ssDNA):

5'- GATTACACAGATATGACC... – 3'
3'- CTAATGTGTCTATACTGG.... – 5'

2.3 Gens

Un gen és el conjunt d'una seqüència determinada de nucleòtids que pot tenir l' informació necessària per sintetitzar una macromolècula com per exemple proteïnes.

2.4 ORF (Open Reading Frame)

ORF o marc de lectura obert, és cada una de les seqüències d'ADN compreses entre un codó inicial ("ATG") i un finalitzador ("TAA", "TGA", "TAG"). En una seqüència d'ADN hi ha 6 possibles sentits en els que pot haver-hi ORF. Donat que els codons agafen 3 nucleòtids, existeixen 3 possibles posicions d'inici per llegir-los de 3 en 3. A aquests 3 s'han d'afegir els de la cadena complementaria.

Així dons, podem trobar els marcs +1, +2, +3, -1, -2, -3.

+1	5'	GAT TAC ACA GAT ATG ACC	3'
+2	5'	GAT TAC ACA GAT ATG ACC	3'
+3	5'	GAT TAC ACA GAT ATG ACC	3'
-1	3'	CTA ATG TGT CTA TAC TGG	5'
-2	3'	CTA ATG TGT CTA TAC TGG	5'
-3	3'	CTA ATG TGT CTA TAC TGG	5'

3 Requeriments

3.1 Requeriments de l'aplicació

En aquest punt detallem quin són els serveis que ha d'oferir el sistema i quines són les seves limitacions.

A les reunions es va determinar que l'objectiu final de l'aplicació, era la realització d'una eina visual integrada en el laboratori virtual (labVir), que oferís als estudiants una representació gràfica de molècules d'ADN. Aquesta eina hauria de permetre representar diferents capes associades a conceptes relacionats amb les molècules.

Es va determinar que aquestes capes serien:

- Gens
- Identificació de segments
- Localització de seqüències
- ORF

Les molècules havien de ser reals obtingudes de terceres bases de dades com GenBank o molècules pròpies creades de zero. El tamany de les molècules es va limitar a 1mpb per poder fer un tractament ràpid i no bloquejar el servidor.

Un cop obtinguda la molècula s'haurien de poder marcar els diferents conceptes, així com poder fer un tractament de crear, modificar i esborrar els diferents elements. Aquests elements es guardarien en una Base de Dades per la seva posterior consulta.

La capa Gens permetria seleccionar els diferents paràmetres que defineixen un gen: com és un nom, on comença i acaba el gen, el marc de lectura, si esta en la cadena simple o en la complementaria o si es una proteïna. Per facilitar l'estudi dels gens era recomanable crear una capa anomenada "Marques" per poder seleccionar zones de la molècules que poguessin ser mostrades conjuntament amb els gens. A la representació també era interessant veure on estaven ubicades les seqüències de restricció, uns patrons que permetrien als biòlegs manipular químicament la molècula. I per últim era molt interessant poder localitzar els ORF de la molècula tractada, per mostrar-los conjuntament amb els gens.

L'aplicació havia de ser en Flash (ActionScript) per poder-la integrar fàcilment amb labVir, i poder-la oferir via web remota o un servidor web local.

3.2 Requeriments de l'usuari

En aquesta secció descriurem els requeriments que necessita un usuari per poder utilitzar el sistema.

Cada alumne es connectarà a l'aplicació mitjançant un ordinador personal amb un servidor de pàgines web. Aquest haurà de tenir els serveis de PHP, PERL i una Base de Dades, preferentment MYSQL. Existeix la possibilitat de treballar amb un servidor web remot on tots els usuaris compartirien les dades.

L'aplicació forma part de labVir per tant necessitarà els mateixos requeriments que s'especifiquen en el projecte que es va desenvolupar.

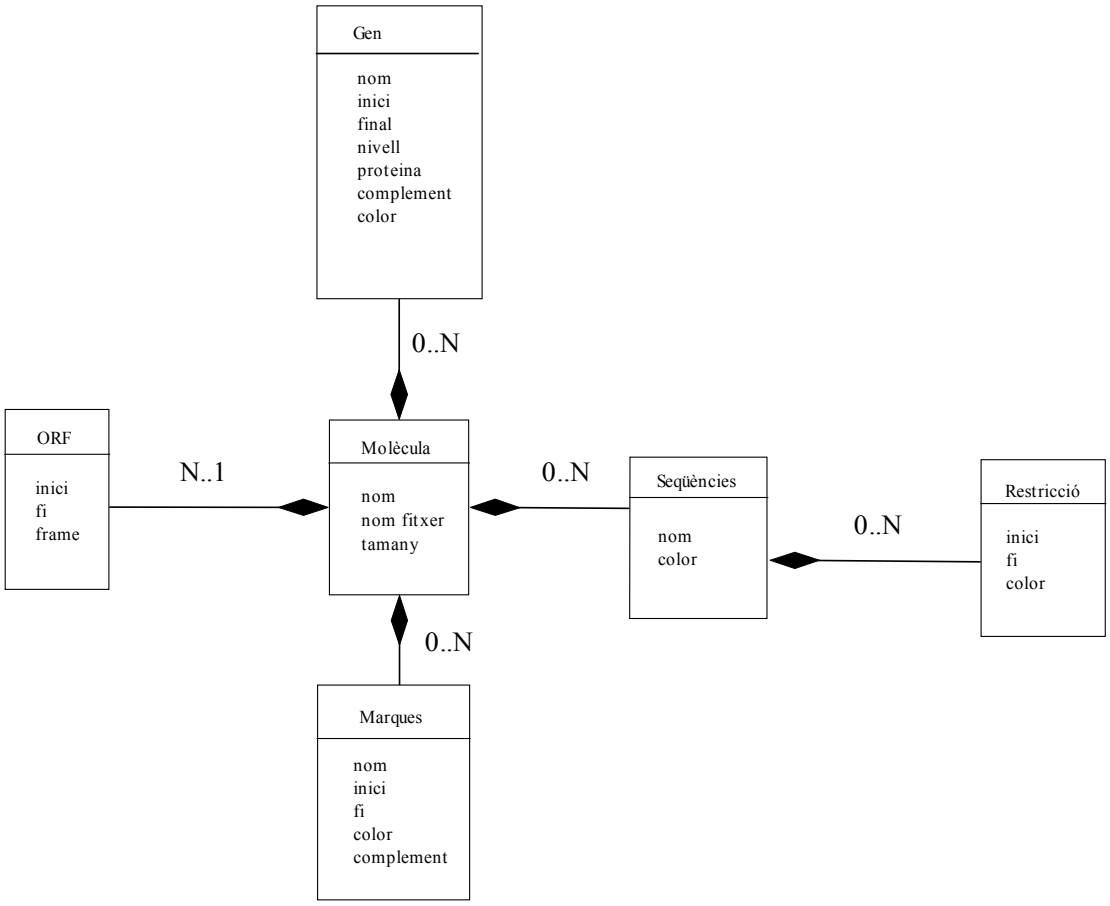
Veiem els passos que haurà de fer l'usuari per accedir a l'aplicació:

1. Per accedir a l'aplicació obriran el navegador i entraran a l'adreça que el professor els indiqui.
2. El sistema demanarà a l'alumne que s'identifiqui amb un nom.
3. Un cop identificat el sistema presentà un llistat de les eines a utilitzar
4. L'alumne triarà l'opció de Visor
5. Carregada l'eina l'alumne podrà començar a treballar amb la molècula creada per defecte o crear una nova.

4 Anàlisi del sistema

Un cop definits els requeriments de l'aplicació i d'usuari procedim a detallar totes les funcionalitats de l'aplicació.
Per fer-ho creem un model de dades orientat a objectes que ens ajudarà a obtenir els elements que formaran part del projecte, la relació entre ells i l'informació que hem d'emmagatzemar.

4.1 Model de dades



4.2 Descripció del model de dades

Molècula

Descripció: guarda informació necessària de les molècules amb les que treballarem

- Nom: nom de la molècula; String
- Nom fitxer: nom del fitxer .fasta de la molècula; String
- Tamany : tamany de la molècula; Integer

Gens

Descripció: guarda informació dels gens que volem marcar

- Nom: nom del gen; String
- Inici: inici del gen; Integer
- Final: final del gen; Integer
- Nivell: marc de lectura on es localitza el gen; Integer
- Proteïna: si el gen genera una proteïna; Boolean
- Complement: si el gen es troba en la cadena simple o en la complementaria; Boolean
- Color: color que li donarem per la seva representació; Integer

Seqüències

Descripció: guarda informació sobre les seqüències de codi que volem localitzar

- Nom: seqüència de nucleòtids a buscar (“GATC”) ; String
- Color: color amb el que els volem representar; Integer

Restriccions

Descripció: guarda totes les posicions on s’ha trobat una seqüència determinada de nucleòtids

- Inici: posició inicial de la seqüència; Integer
- Final: posició final de la seqüència; Integer
- Color: color per representar la seqüència; Integer

Marques

Descripció: guarda informació sobre seccions del codi que volem visualitzar independentment del codi ADN que hi hagi, guardant només informació d’inici i final.

- Nom: nom de la marca; String
- Inici: posició inicial; Integer
- Final: posició final de la marca; Integer
- Color: color que li volem donar a la marca: Integer
- Complement: ubicació de la marca en la cadena simple o complementaria; Boolean

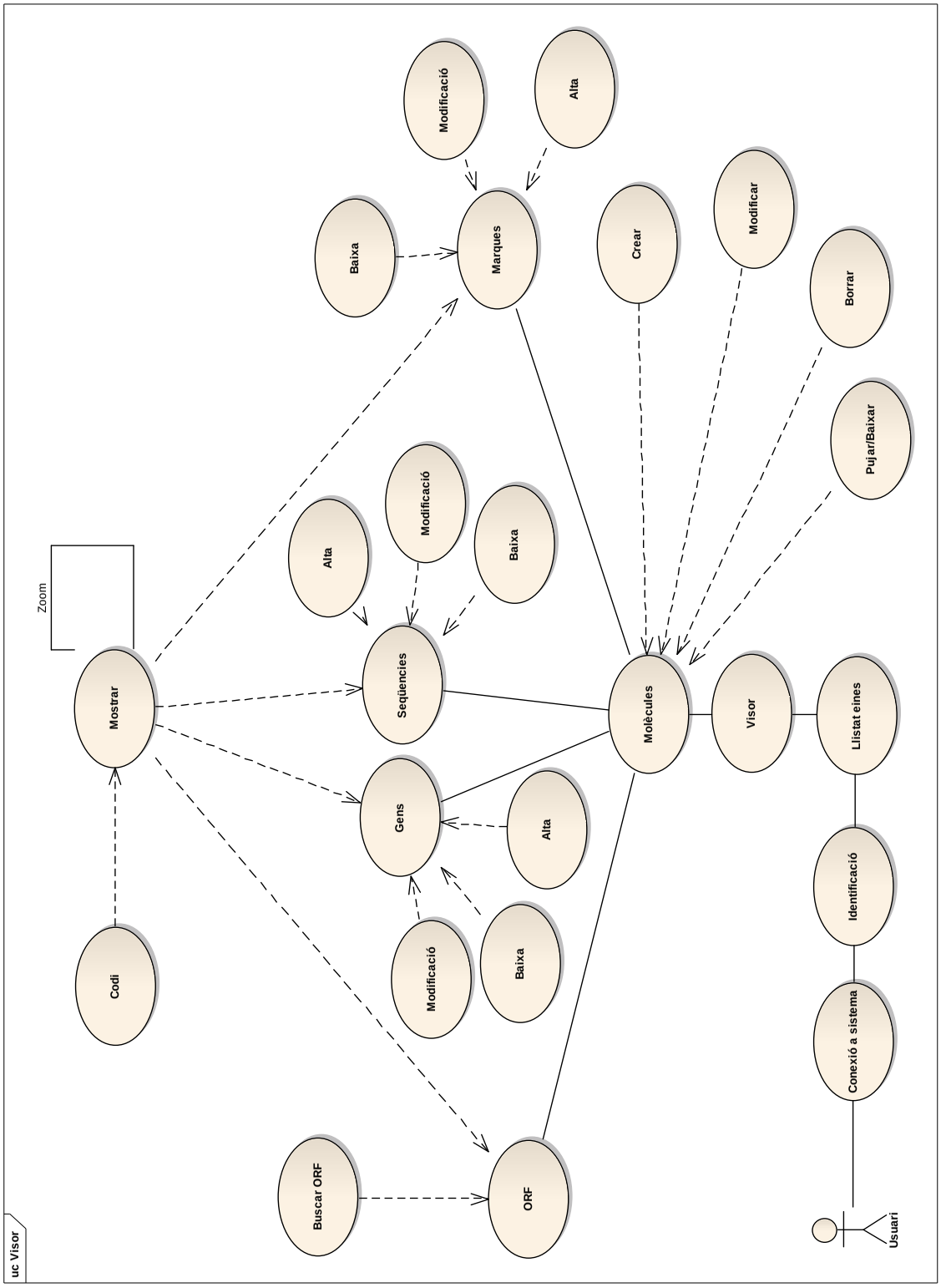
ORF

Descripció: guardarà informació de les ORF calculades per a cada molècula

- Inici: posició inicial del ORF; Integer
- Final: posició final del ORF; Integer
- Frame: marc de lectura on s'ha trobat l' ORF; Integer

4.3 Esdeveniments

Per poder determinar quines funcions tindrà el sistema, utilitzarem un diagrama d'esdeveniments, que ens ajudarà a veure quines accions s'hauran de realitzar.



4.4 Descripció dels esdeveniments

Alta Gen
Tipus: Informació.
Resposta: Es registra el nou gen a la Base de dades.
Descripció: Donar d'alta a la base de dades el nou gen. Regenerar l'arbre XML associat a la molècula.

Modificació Gen
Tipus: Informació.
Resposta: Modificar el gen a la Base de dades.
Descripció: Modifica el gen entrat a la base de dades. Regenerar l'arbre XML associat a la molècula.

Baixa Gen
Tipus: Informació.
Resposta: Dóna de baixa el gen a la Base de dades.
Descripció: Elimina el gen entrat a la base de dades. Regenerar l'arbre XML associat a la molècula.

Visualitzar / Amagar Gen
Tipus: Visualització
Resposta: Mostra o amaga la representació gràfica corresponent a l'informació del gen.
Descripció: Crea un element gràfic representatiu d'un gen, del tamany i en el marc de lectura corresponent, situant-lo en funció dels seus paràmetres (inici, final, cadena simple o complementaria, color i marc de lectura), dins la representació de la molècula. Si l'element s'està mostrant, el destrueix.

Alta Seqüència
Tipus: Informació.
Resposta: Es dóna d'alta a la nova seqüència a la base de dades.
Descripció: Es dóna d'alta la seqüència a la base de dades i es busca aquella seqüència en el codi de la molècula que estem tractant, retornant un llistat de posicions on s'ha localitzat. Aquest llistat es guardarà en una nova taula de la base de dades. Regenerar l'arbre XML associat a la molècula.

Modificació Seqüència
Tipus: Informació.
Resposta: Es modifica la seqüència a la base de dades.

Descripció: Es modifica la seqüència a la base de dades i es busca aquella seqüència al codi de la molècula que estem tractant. S'esborra l'antic llistat de posicions i es genera un nou llistat. Regenerar l'arbre XML associat a la molècula.

Baixa Seqüència
Tipus: Informació.
Resposta: Dóna de baixa la seqüència a la base de dades.
Descripció: S'elimina la seqüència i les posicions on s'ha localitzat de la base de dades. Regenerar l'arbre XML associat a la molècula.

Visualitzar / Amagar Seqüència
Tipus: Visualització
Resposta: Mostra o amaga la representació gràfica corresponent a la informació de la seqüència.
Descripció: Crea un element gràfic representatiu de la ubicació i tamany de la seqüència, situant-lo en funció dels seus paràmetres (inici, final i color), dins la representació de la molècula. Si l'element s'està mostrant el destrueix.

Alta Marca
Tipus: Informació
Resposta: Registra una nova marca a la base de dades.
Descripció: Es dóna d'alta una nova marca a la base de dades. Regenerar l'arbre XML associat a la molècula.

Modificació Marca
Tipus: Informació
Resposta: Es modifica les dades de la marca en la base de dades
Descripció: Modifica la marca en la base de dades. Regenerar l'arbre XML associat a la molècula.

Baixa Marca
Tipus: Informació
Resposta: Dóna de baixa la marca a la base de dades.
Descripció: Elimina el registre de la marca a la base de dades. Regenerar l'arbre XML associat a la molècula.

Visualitzar / Amagar Marca
Tipus: Visualització
Resposta: Mostra o amaga la representació gràfica corresponent a la informació de la marca.

Descripció: Crea un element gràfic representatiu d'una marca, situant-lo, en funció dels seus paràmetres (inici, final, color i cadena simple o complementaria), dins la representació de la molècula. Si l'element s'està mostrant, el destrueix.

Buscar ORF
Tipus: Informació
Resposta: Busca patrons ORF a la seqüència de la molècula.
Descripció: Busca els patrons d'ORF al codi de la molècula i es guarda a la base de dades. Regenerar l'arbre XML associat a la molècula.

Visualitzar / Amagar ORF
Tipus: Visualització
Resposta: Mostra o amaga la representació gràfica corresponent a la informació del ORF.
Descripció: Crea un element gràfic representatiu d'un ORF, situant-lo en funció dels seus paràmetres (inici, final, marc de lectura, color i cadena simple o complementaria), dins la representació de la molècula. Si l'element s'està mostrant el destrueix.

Crear Molècula
Tipus: Informació
Resposta: Crea una nova molècula a la base de dades.
Descripció: Crea una nova molècula a la base de dades associada a un fitxer de tipus ".fasta". Regenerar l'arbre XML de molècules. Regenerar l'arbre XML associat a la molècula.

Modificar Molècula
Tipus: Informació
Resposta: Modifica l' informació de la molècula a la base de dades així com el codi de seqüència de la molècula si ha estat modificat.
Descripció: Modifica els paràmetres de la molècula existent a la base de dades i del codi corresponent de la seqüència si ha estat modificada. Regenerar l'arbre XML de molècules. Regenerar l'arbre XML associat a la molècula.

Eliminar Molècula
Tipus: Informació
Resposta: Elimina la molècula i tots els elements relacionats a la base de dades.
Descripció: Elimina de la base de dades la molècula donada i tots els elements relacionats amb aquesta (gens, marques, seqüències i ORF). Regenerar l'arbre XML de molècules. Regenerar l'arbre XML associat a la molècula.

Pujar/Baixar Molècula
Tipus: Informació
Resposta: Puja o descarrega arxius ".fasta" que contenen el codi ADN de la molècula.

Descripció: Permet pujar i baixar fitxers d'extensió ".fasta" per relacionar-los posteriorment amb les molècules creades a la base de dades.

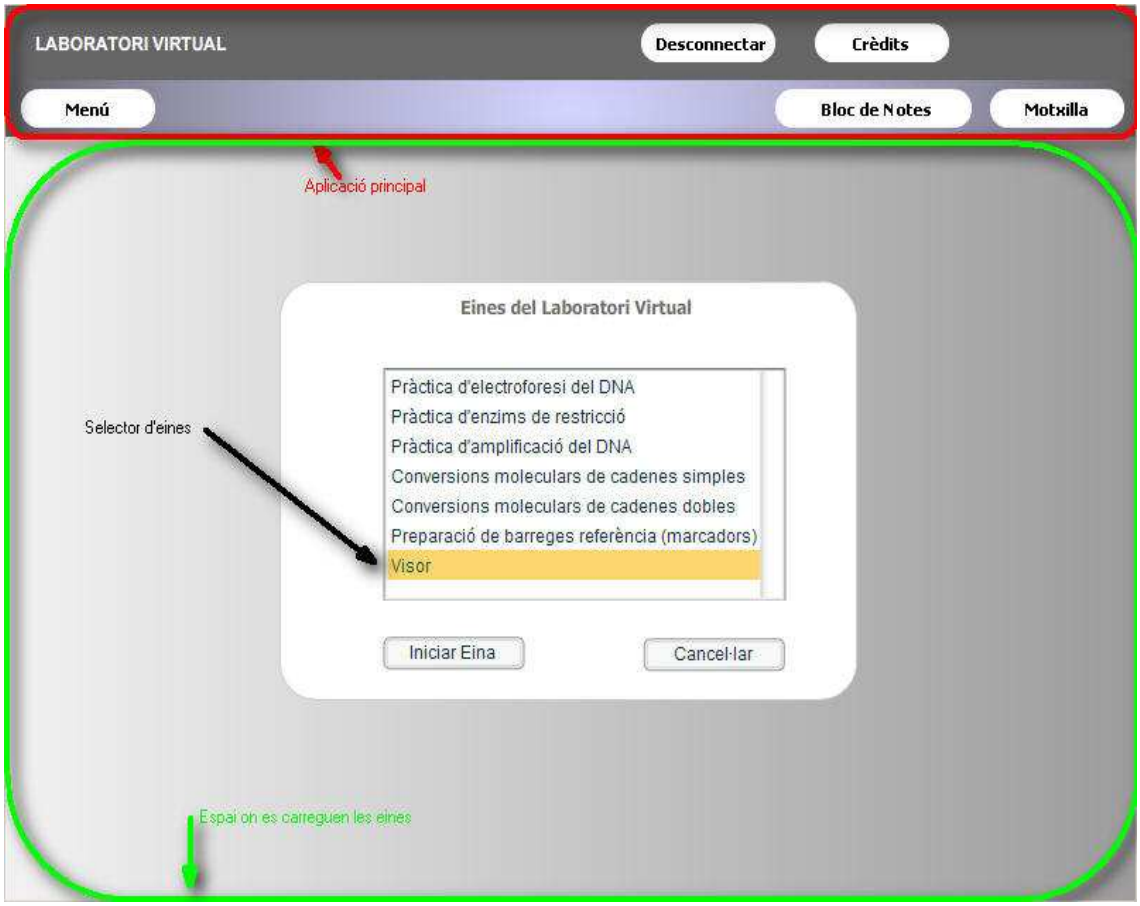
Zoom de la Molècula
Tipus: Visualització
Resposta: Fa una ampliació de la representació gràfica del elements mostrats.
Descripció: Donat un tamany i posició de zoom busca els elements que s'estan mostrant. Si es localitzen dins del rang de zoom, es creen elements equivalents amb tamany ampliat. Els elements visibles fora de rang s'eliminen de la representació del zoom.

Codi de la Molècula
Tipus: Informació
Resposta: Retorna la seqüència d'ADN corresponent.
Descripció: Donada una posició es busca la seqüència d'ADN corresponent i es mostra.

5 Disseny del Sistema

5.1 Arquitectura

En el disseny de l'aplicació partíem d'un entorn ja creat en un anterior projecte anomenat labVir. La idea era aprofitar aquest sistema ja existent. Es composava d'un programa principal que carregava dinàmicament "eines" o pràctiques creades específicament per treballar amb l'entorn. Aquestes eines realitzaven funcions especifiques relacionades amb la biologia.



L'entorn estava realitzat en Flash i el codi ActionScript. Pel seu desenvolupament s'havia utilitzat la plataforma AriWare RIA Plataform versió 2.02. Plataforma open-source que proposa la separació del codi de la part gràfica i multimèdia, fent ús d'una estructura orientada a objectes basada en esdeveniments.

La separació de la part visual de la programació, facilita la feina a l'hora de mantenir l'aplicació o, com en el cas que ens pertoca, entendre l'estructura creada sense haver de localitzar les parts de codi pels diversos objectes Flash.

Aquesta decisió presa en el desenvolupament de labVir ha permès que de manera més o menys costosa es puguin crear noves eines sense interferir en les ja creades.

A continuació veurem un diagrama de classes de l'entorn labVir on es veuen les diferents parts que el composen i com enllaça amb PHP.

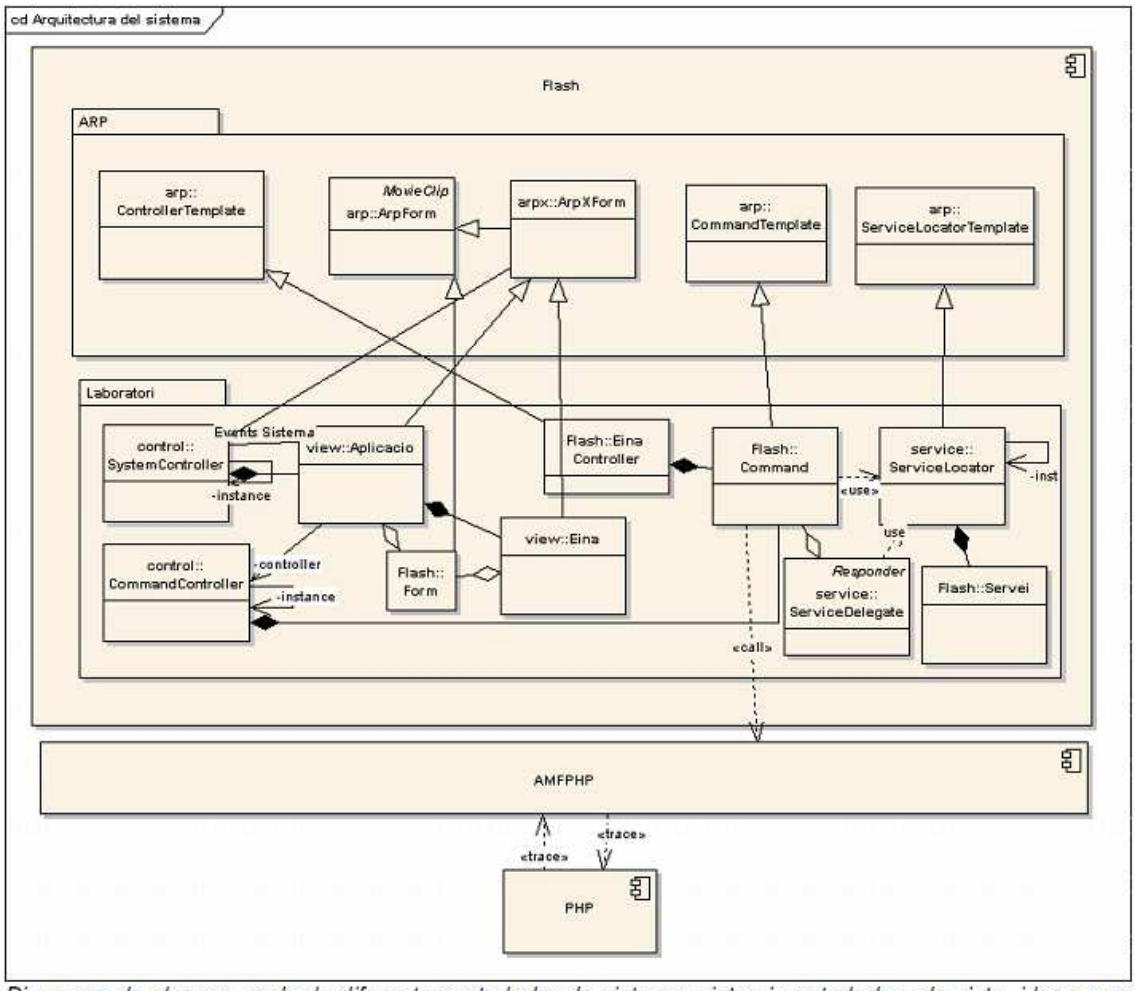


Diagrama de classes, amb els diferents controlador de sistema, vistes i controladors de vista, i les seves relacions. Es basa amb la plataforma ARP i l'extensió ARPX (SystemController i ArpXForm)

Així des de l'inici es tenia clar que havia de ser una eina integrada en aquest entorn. Aquest fet ens implicava una sèrie de pautes com ara la realització en Flash i la separació del codi i el disseny gràfic.

L'aplicació havia de mostrar diferents conceptes relacionats amb molècules d'ADN. Aquests conceptes o dades s'havien d'emmagatzemar d'alguna forma per evitar haver-les d'entrar cada vegada. En aquest punt es va decidir utilitzar una base de dades per guardar aquesta informació. MYSQL va ser l'opció triada per realitzar aquesta acció per varis motius, és gratuïta i al projecte inicial "labVir" ja es va proposar. Per fer més ràpida l'aplicació i evitar realitzar constantment consultes a la base de dades es tria l'opció de generar fitxers XML amb les dades de la base de dades. Aquests fitxers són molt utilitzats en diversos components de Flash que ens anirien bé per l'aplicació. També ofereixen un ús "offline" de l'aplicació, limitant-la només a la visualització de les dades ja existents. Per

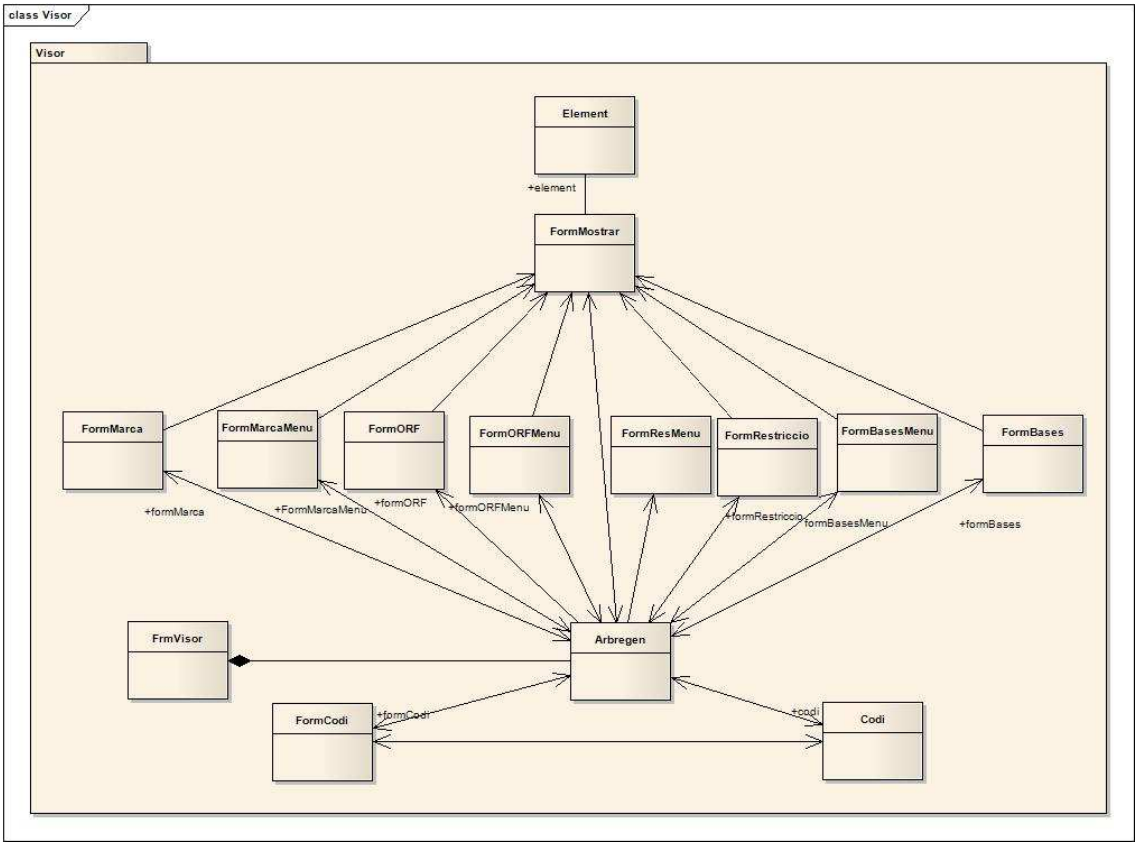
enllaçar Flash amb la base de dades s'utilitza un servei ja present en labVir, anomenat AMFPHP; que realitza la funció d'intercanvi de dades entre ActionScript i PHP.
L'aplicació havia de realitzar càlculs com la cerca d'ORF o tractament de fitxers de tamanys considerables, que feien inviable l'ús de PHP o ActionScript per processar-los. Utilitzar algun d'aquests llenguatges pel tractament podria implicar el bloqueig del servidor (per el PHP) o del navegador (FLASH/ActionScript). Per evitar aquesta situació es va triar utilitzar Perl pel tractament de processos de càlcul i fitxers.

Utilitzar Perl oferia varies avantatges:

- Llenguatge àmpliament utilitzat en bioinformàtica.
- Es poden executar en segon pla, evitant bloquejar el processador.
- Independència de la resta d'aplicació (facilita la modificació o millora sense refer tota l' aplicació).
- Major velocitat de càlcul respecte les opcions en PHP i ActionScript.

La separació d'aquesta part de la resta de l'aplicació, implica la independència del llenguatge utilitzat per fer els càlculs, oferint així la possibilitat de realitzar-los amb llenguatges més potents, com per exemple C. Per fer-ho, només caldria retornar les dades en el mateix format que les aplicacions creades en aquest projecte.

5.2 Diagrama de classes simplificat



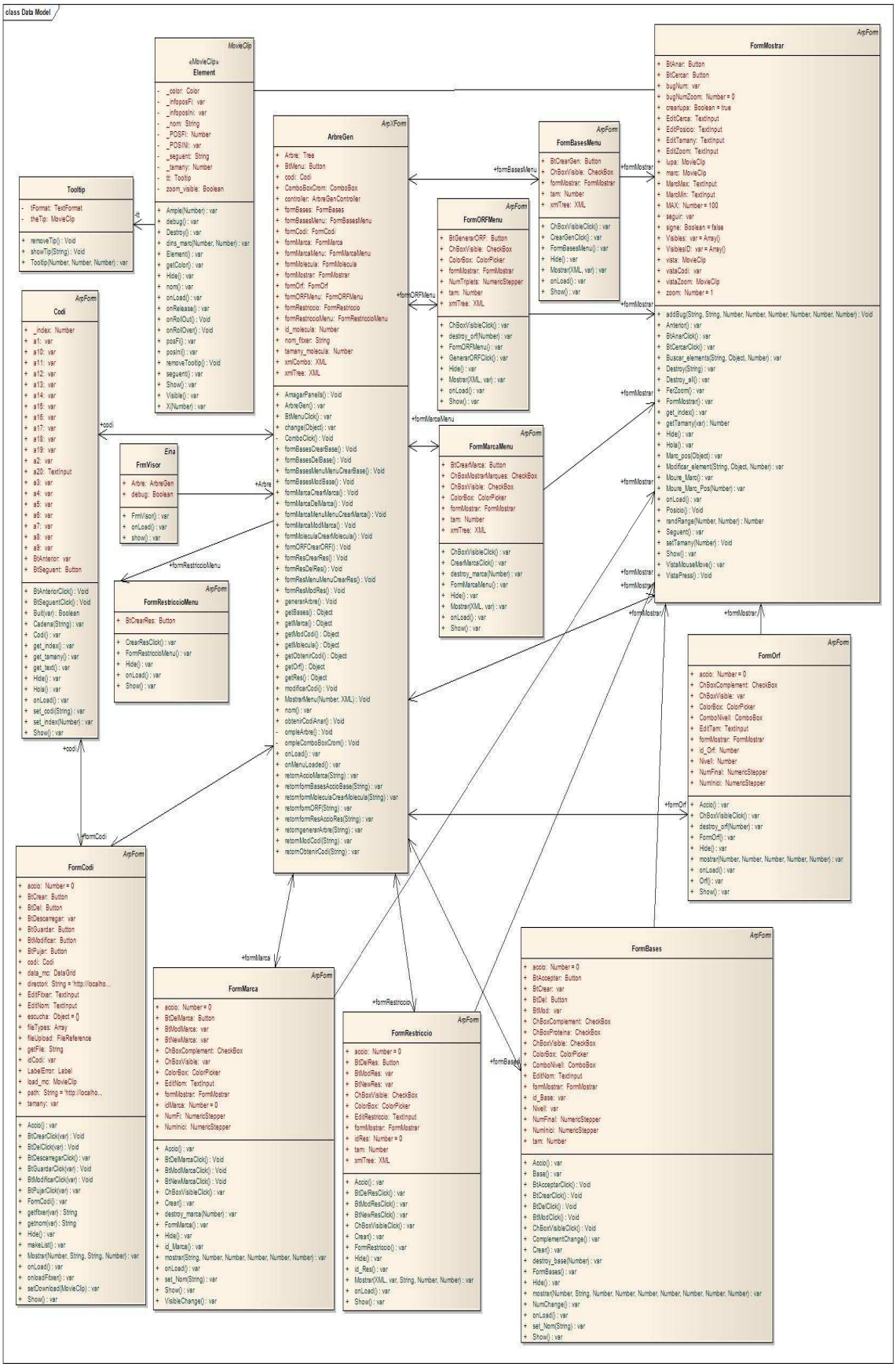
Al diagrama podem observar les diferents classes que han aparegut respecte l'anàlisi. La majoria són classes intermediàries de menús entre les classes principals que sortien en l'anàlisi i la representació gràfica.
En aquest grup de classes noves trobem els FormMarcaMenu, FormBasesMenu, FormOrfMenu que són interfícies gràfiques que carreguen les opcions que es poden fer amb les classes principals.

La traducció directe del diagrama de l'anàlisi seria:

- Gens: FormBases
- Seqüències: FormResMenu
- Restriccions: FormRes
- Marques: FormMarca
- ORF: FormORF
- Molècula: FormCodi

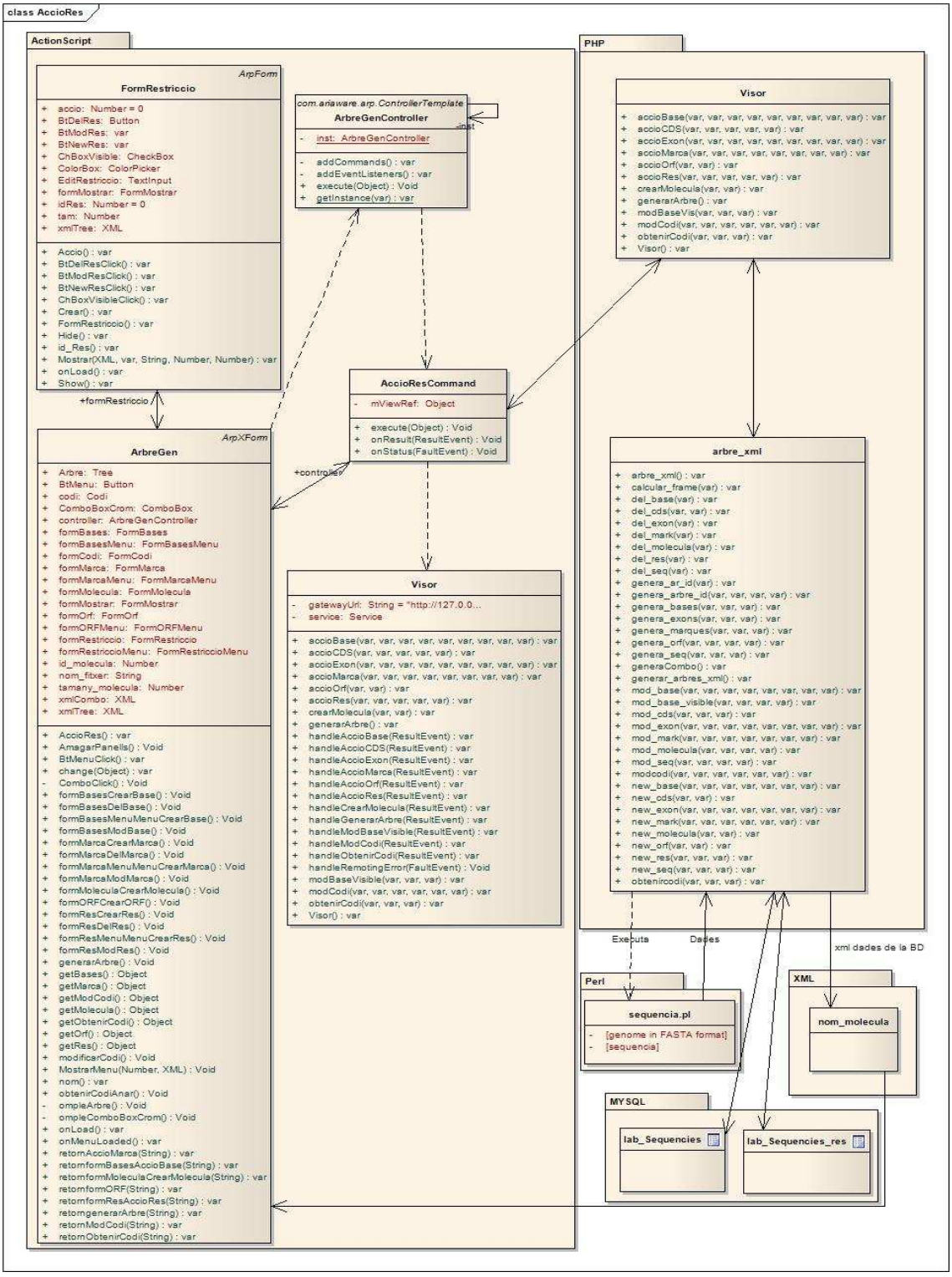
Veiem que apareix una classe *"Element"*, que correspon als elements que es creen a la visualització. La classe *"Codi"* que s'encarregarà de mostrar el codi ADN. La classe *"ArbreGen"* que és l'encarregada de gestionar totes les interfícies i crides al sistema i enllaça amb AMFPHP. FormVisor que és l'eina que enllaça amb labVir i crea l'entorn.

5.3 Diagrama de classes ampliat



5.4 Disseny entorn ActionScript / PHP / Perl / BD / XML

El sistema enllaça l'aplicació creada en Flash amb una Base de Dades i la execució d'aplicacions externes en Perl. Tot seguit, veurem en un esquema complet tot el recorregut que es realitza.



Al diagrama veiem les classes que es fan servir per les accions amb seqüències.

Començant amb el *"FormRestriccio"* , corresponent a la interfície gràfica de seqüències (la veurem en el disseny d'interfícies). *"FormRestriccio"* provocaria un event (Crear/Modificar/Esborrar) que captura *"ArbreGen"*. Aquest recolliria la informació necessària i l'acció a realitzar del *"FormRestriccio"*, provocant un event al sistema.
Els events del sistema ARPAriware estan definits en fitxers de controladors. En el cas de la nostra aplicació, s'ha crear un fitxer propi (*ArbreGenControler.as*) integrat als controladors de labVir.

(com.laboratori.control.ArbreGenControler.as, linea 21)

```
private function addEventListeners (){
	...
	app.addEventListener ( "accioRes", this );
	app.addEventListener ( "accioMarca", this );
	...
}

private function addCommands (){
	trace ("INFO ArbreGenController::addCommands()");
	addCommand ( "accioOrfCommand", AccioOrfCommand );
	addCommand ( "accioBaseCommand", AccioBaseCommand );
	addCommand ( "accioResCommand", AccioResCommand );
	addCommand ( "accioMarcaCommand", AccioMarcaCommand );
	...
}
```

En aquest fitxer estan definits els events del sistema i a quin servei estan associats. A l'exemple anterior veiem que el controlador carrega el **Command** *"AccioResCommand "* que utilitza els servei definits en *(com.laboratori.service.Visor.as)* i associats als serveis PHP del fitxer (*amfphp.services.practiques.visor.php*). Conegut el servei ja pot intercanviar informació amb el PHP.

(com.laboratori.command.visor.AccioResCommand.as)

```
import mx.remoting.*;
import mx.rpc.*;
import mx.utils.Delegate;
import com.laboratori.service.ServiceLocator;

class com.laboratori.command.visor.AccioResCommand
{
	private var mViewRef:Object;

	public function execute(pViewRef:Object):Void
	{
		trace("AccioResCommand::execute");
		mViewRef = pViewRef;
		var vo:Object = mViewRef.getRes();
		var service:Service = ServiceLocator.getInstance().getService('Visor');
		var pc:PendingCall =
service.accioRes(vo.id_crom,vo.id_res,vo.res,vo.color,vo.accio);
		pc.responder = new RelayResponder(this, "onResult", "onStatus");
	}

	public function onResult(re:ResultEvent):Void
	{
```

```
                trace("AccioResCommand::onResult");
                //Actualitzo el formulari amb les dades del resultat
                mViewRef.retornformResAccioRes(re.result);
        }

        public function onStatus(fe:FaultEvent):Void
        {
                trace("AccioResCommand::onStatus");
                throw new Error ("Command failed: " + fe.fault.description);
        }
}
```

(com.laboratori.service.Visor.as, varis fragments)

```
import mx.remoting.*;
import mx.rpc.*;
import mx.utils.Delegate;
import mx.remoting.debug.NetDebug;

class practiques.Visor
{
        //Change the gateway URL as needed
        private var gatewayUrl:String = "http://127.0.0.1/lab/amfphp/gateway.php";
        private var service:Service;

        function Visor()
        {
                NetDebug.initialize();
                this.service = new Service(this.gatewayUrl, null, "practiques.Visor");

        }

        //Genera el menú xml desde BD.
        function generarArbre()
        {
                var pc:PendingCall = service.generarArbre();
                pc.responder    =    new    RelayResponder(this,    "handleGenerarArbre",
"handleRemotingError");
        }

...
        //Crea una nova seqüencia de restriccio en la BD.
        function accioRes(id_crom, id_seq, res, color, accio)
        {
                var   pc:PendingCall   =   service.accioRes(id_crom,   id_seq,   res,   color,
accio);
                pc.responder     =     new     RelayResponder(this,     "handleAccioRes",
"handleRemotingError");
        }

...
        function handleRemotingError( fault:FaultEvent ):Void
        {
        NetDebug.trace({level:"None", message:"Error: " + fault.fault.faultstring });
        }
}
```

(amfphp.services.practiques.visor.php, varies linies)

```
<?php

include('../../../config.php');
include('../../../configuracio/arbre_xml.php');
include(INCLUDE_ROOT.'/XPath.class.php');

class Visor{
        function Visor(){
                $this->methodTable = array(
                        "generarArbre" => array(
                                "description" => "Genera el menú xml desde BD.",
                                "arguments" => array(),
                                "access" => "remote"
                        ),
...
                        "accioRes" => array(
                                "description" => "Crea una nova Sequencia en la BD.",
```

```
                                "arguments"    =>    array("id_crom","id_seq","res","color",
"accio"),
                                "access" => "remote"
                        )
                );
        }
...
        function accioRes($id_crom,$id_seq,$res,$color,$accio) {
                $arbre= new arbre_xml();
                if ($accio==1) {
                        $retorn=$arbre->new_seq($id_crom,strtoupper($res),$color);
                        $arbre->genera_ar_id($id_crom); //Genera el xml nou
                        return $retorn;
                } elseif ($accio==2) {
                        $retorn=$arbre->mod_seq($id_seq,$id_crom,strtoupper($res),$color);
                        $arbre->genera_ar_id($id_crom); //Genera el xml nou
                        return $retorn;
                } elseif ($accio==3) {
                        $retorn=$arbre->del_seq($id_seq);
                        $arbre->genera_ar_id($id_crom); //Genera el xml nou
                        return $retorn;
                } else {
                        return "Error AccioRes";
                }
        }
...
```

Un cop tenim la informació en PHP es realitzen les consultes a la Base de Dades, Perl i fitxers XML (això ho veurem a la fase **Implementació**).
La captura de dades generades en PHP pot ser de dues formes. La primera carregant la informació generada als fitxes XML des dels formularis de Flash, com és el cas de tota la informació sobre la molècula i els seus elements. La segona, és capturant el retorn de PHP i passant-lo al **Command** que ha realitzat la petició, aquest el passarà a *"ArbreGen"* i aquest últim al formulari pertinent. Un exemple seria el retorn de fragments de codi ADN per visualitzar-lo.

Veiem la resta d'events amb diagrames simplificats.

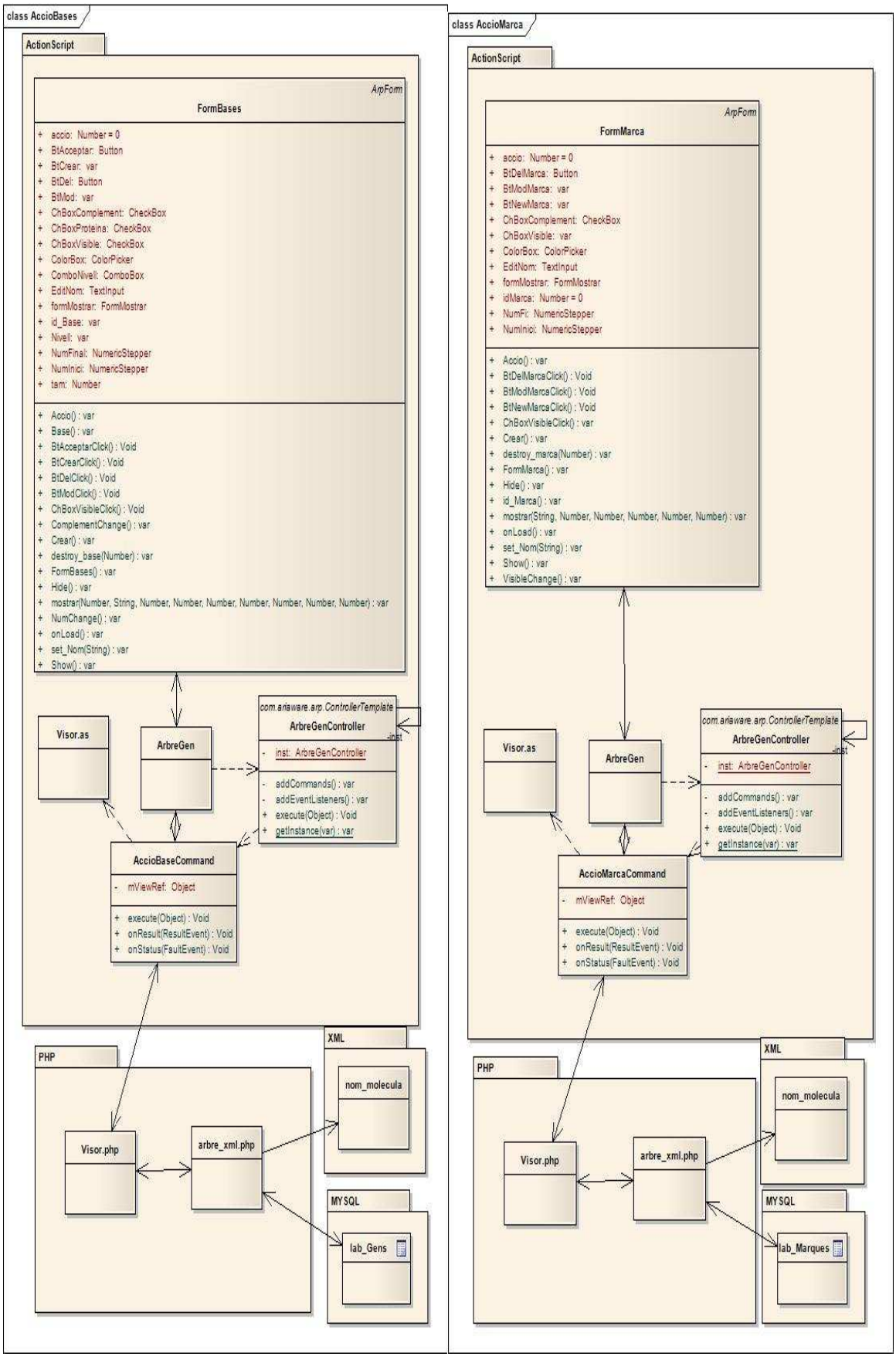
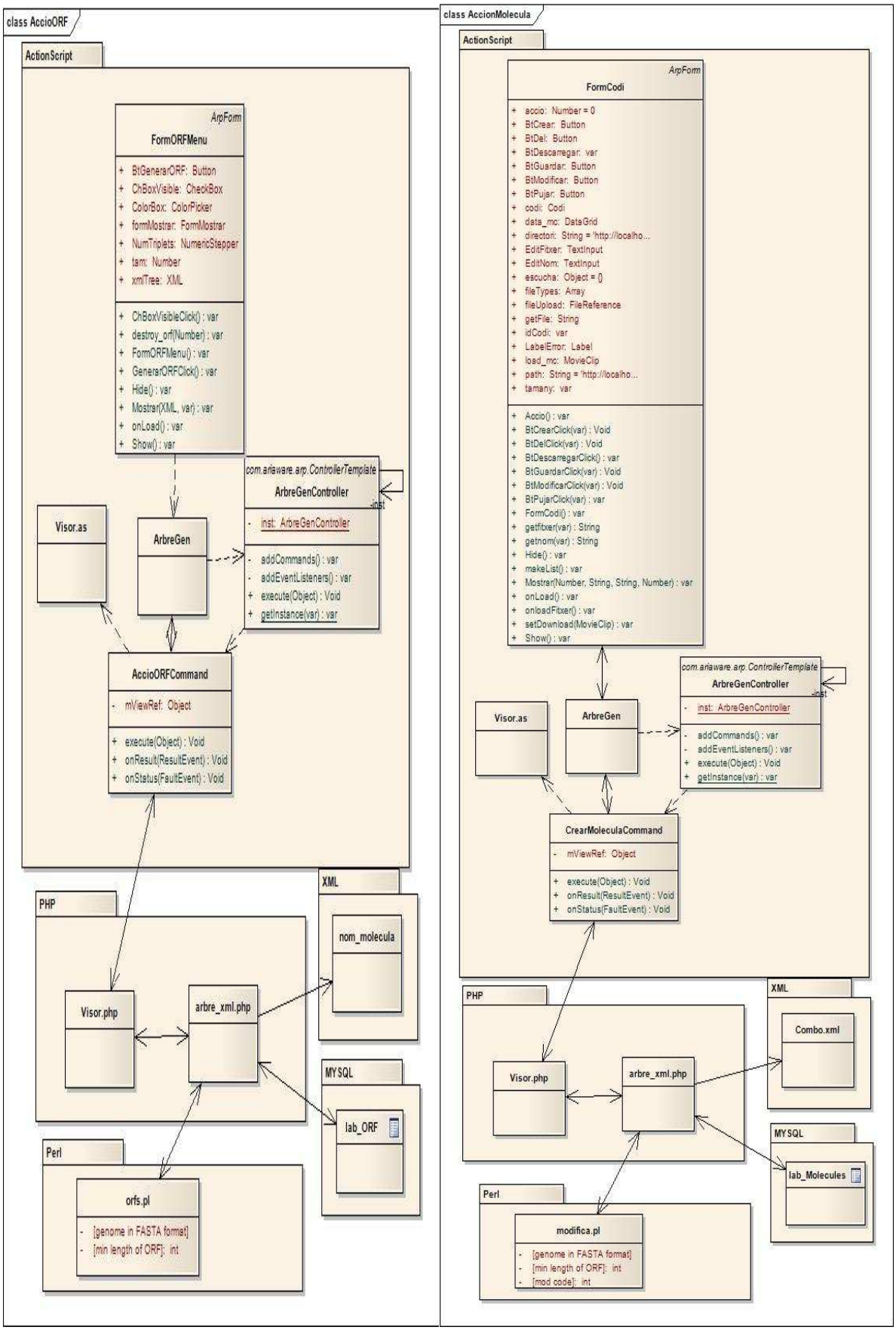
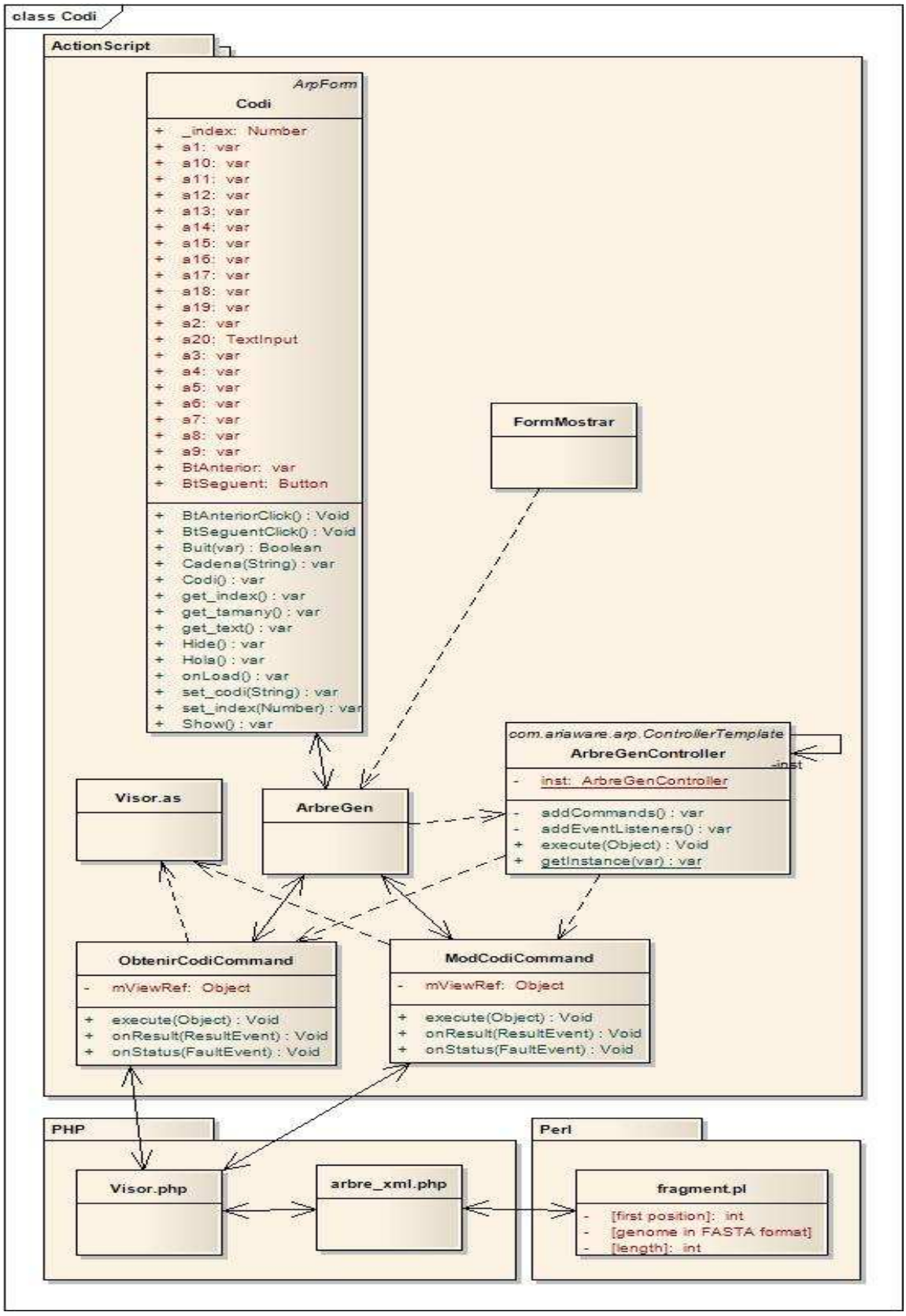


Diagrama de les classes que intervenen a les accions de Gens i Marques.



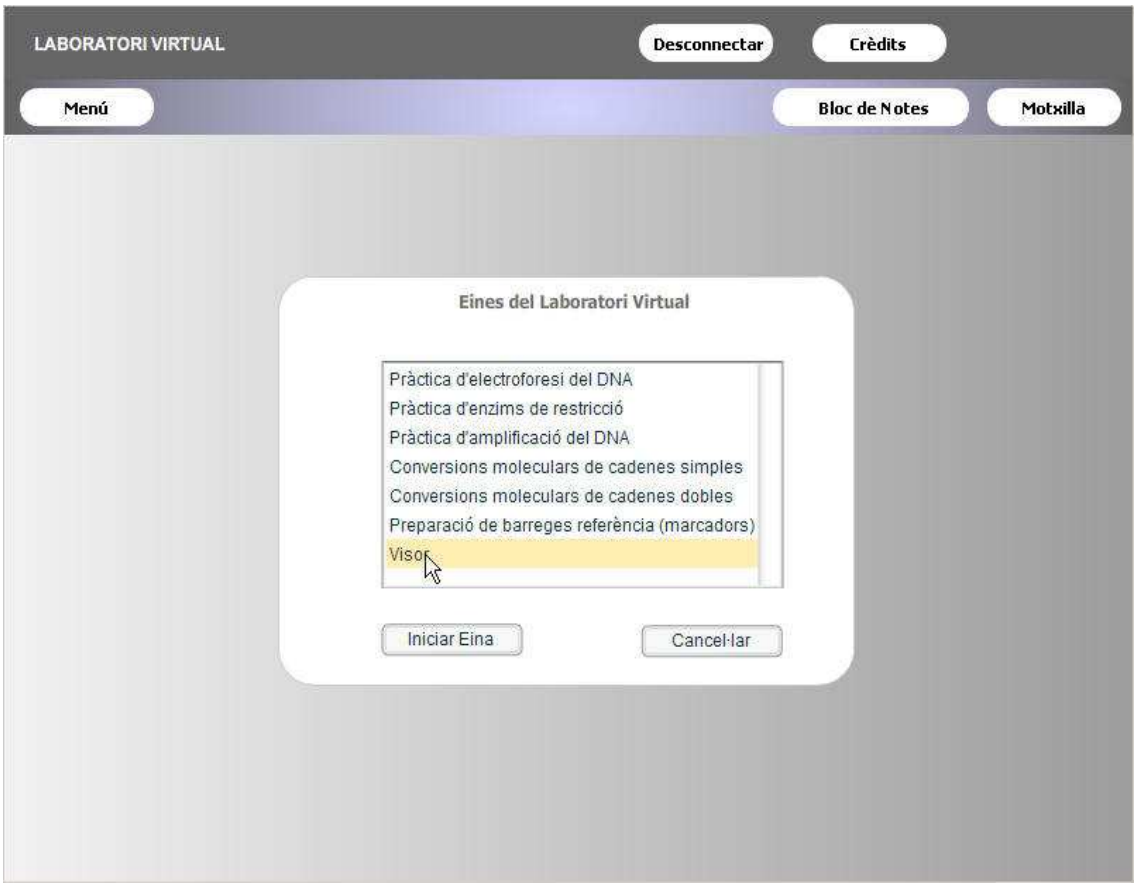
Diagrames de les accions de ORF i molècules.



L'obtenció del codi de AND és pot fer clicant en la pantalla de representació (*"FormMostrar"*) o amb els botons Anterior i Següent del Codi

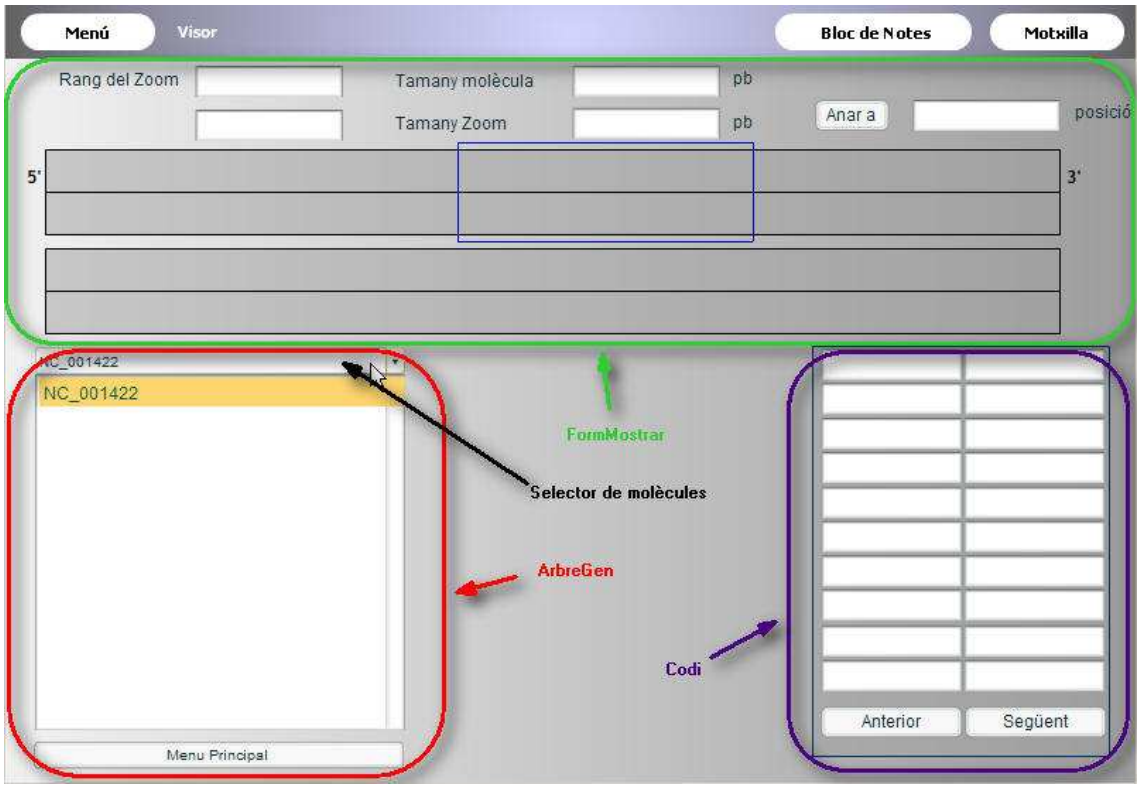
5.5 Disseny d'interfícies gràfiques

L'aplicació desenvolupada està integrada a l'entorn labVir com a una eina més del sistema. A la pantalla de selecció d'eines veiem les diferents pràctiques o eines existents. Per poder començar a treballar amb elles, només hem de triar la que volem carregar i donar al botó de "Iniciar eina".



En qualsevol moment de l'execució de l'aplicació podem retornar aquest menú mitjançant el botó menú de l'entorn labVir. Disposem de les eines pròpies del laboratori, com el bloc de notes que ens permet tenir un lloc ràpid per fer anotacions.

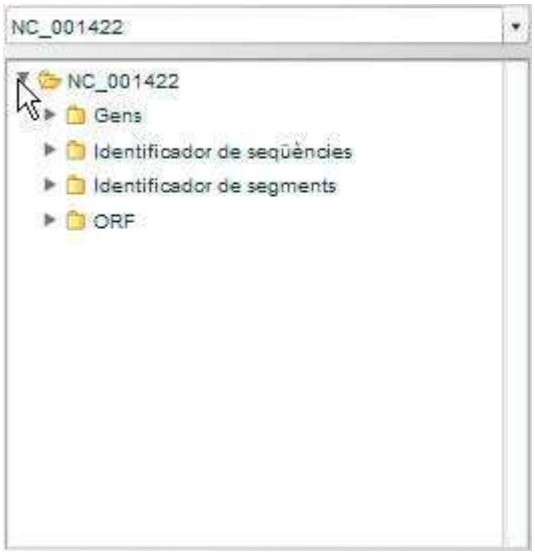
L'eina desenvolupada ens presenta la següent interfície gràfica.



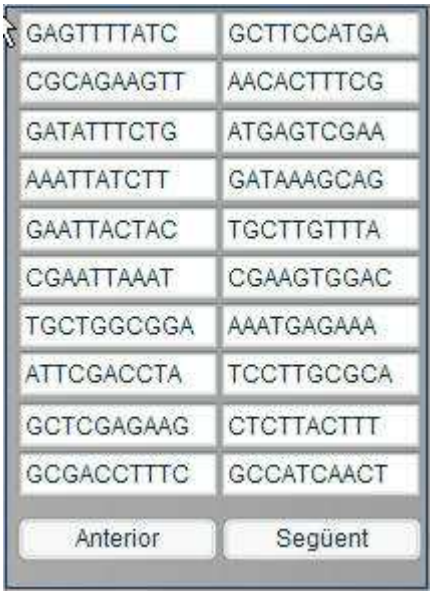
Hem remarcat en diferent colors les diferents parts que la composen.

Així trobem:

- ArbreGen: encarregat de la selecció de molècules creades, i la visualització dels diferents menús que formen part del sistema. Un cop seleccionada la molècula ens carrega el fitxer XML corresponent a aquesta i el menú ens mostra les diferents opcions: visualitzar gens, cercar seqüències, marcar zones o trobar els ORF.

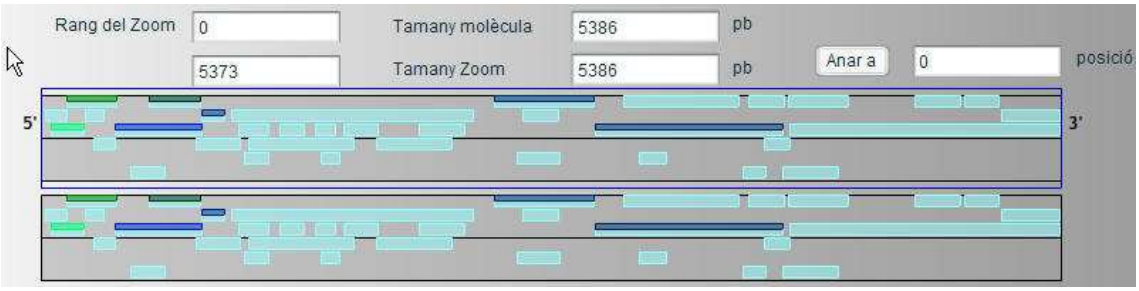


- Codi: encarregat de la representació dels fragments de seqüències de l'ADN de la molècula que estudiem. Agafant com a origen la posició en FormMostrar.



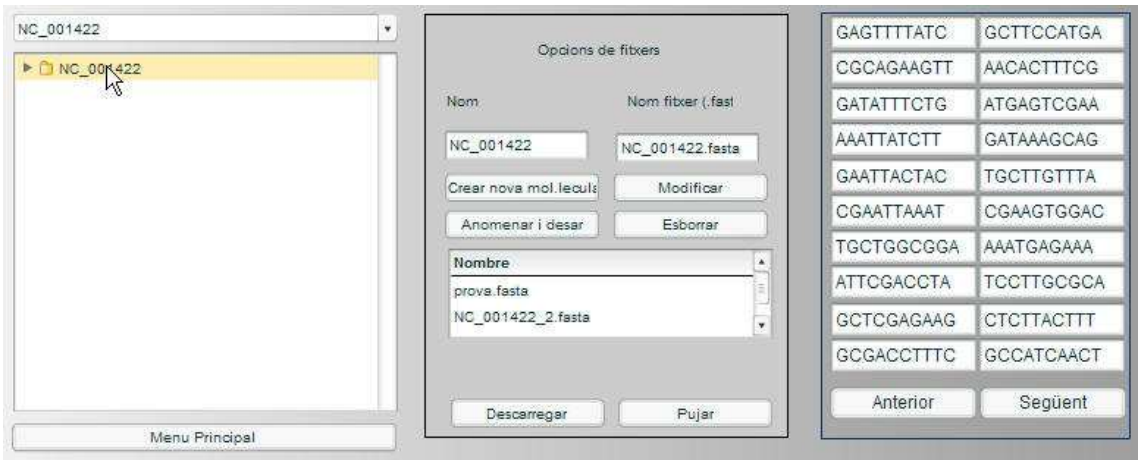
- FormMostrar: encarregat de la visualització dels elements. Permet fer zoom de la representació principal. Aquest apartat es composa de dues parts, la superior que representa la molècula en la seva totalitat i l'inferior que és un zoom d'una zona definida pel tamany del zoom. Les dues parts a la vegada estan dividides en dues zones la superior i l'inferior, simbolitzant la cadena simple o complementaria de l'ADN.

Exemple de representació d'elements



5.5.1 Tractament de molècules

Per poder fer el tractament de les molècules primer hem de tenir-ne una entrada al sistema. Per defecte porta la molècula NC_001422. Per treballar amb altres molècules el primer que hem de fer és crear-la a la base de dades i associar-li un fitxer amb el seu ADN.
Per fer-ho disposem de dues entrades, amb el botó “Menú Principal” o seleccionant una ja existent.

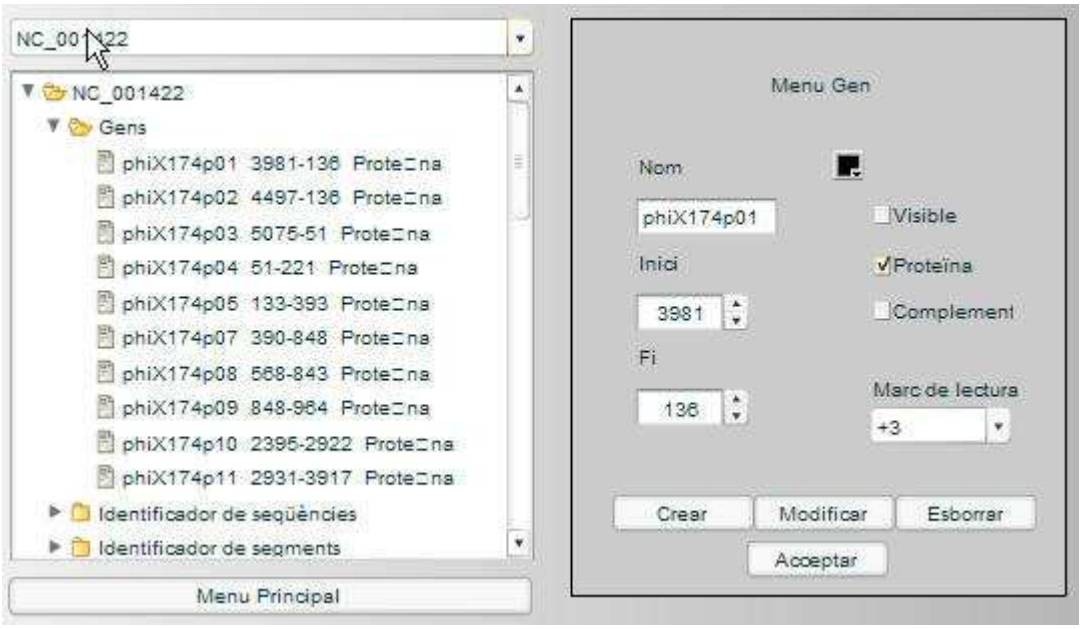


Amb qualsevol de les dues opcions ens apareixerà el menú *FormCodi.* Aquest menú gestiona la creació, modificació i eliminació de molècules del sistema.

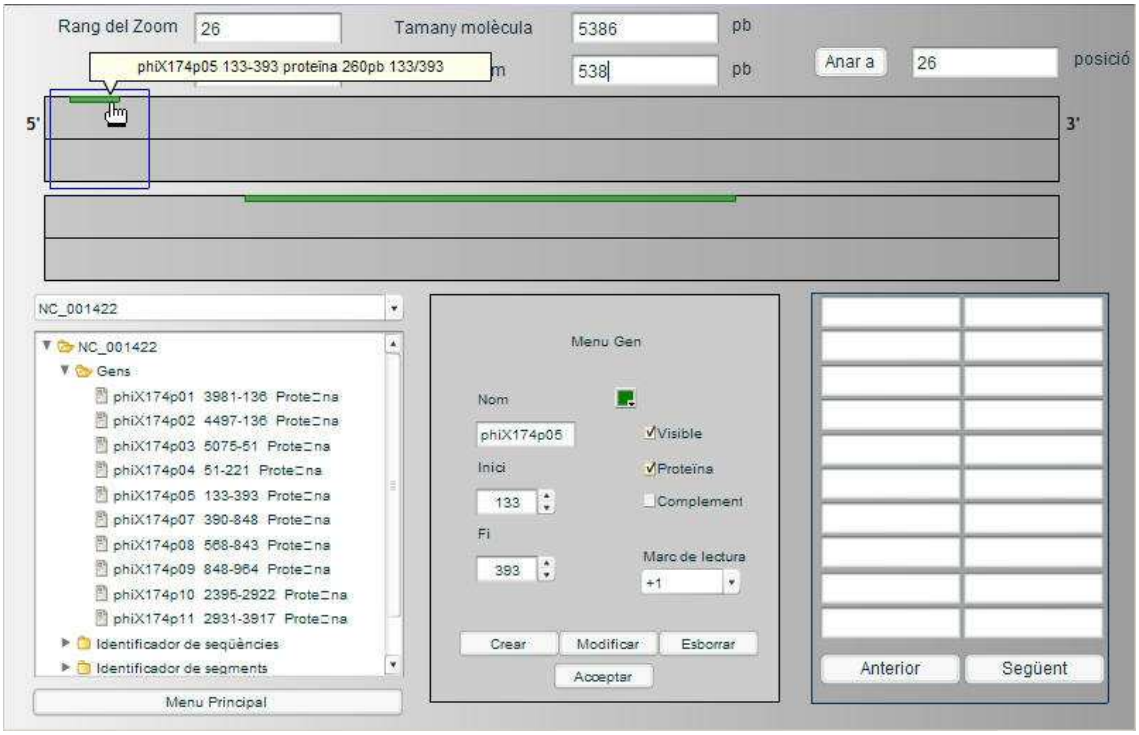
L’opció “modificar”, modifica tant l’entrada a la base de dades de la molècula, com el codi d’aquesta si s’ha modificat a la representació en l’apartat de Codi.
Per assignar un fitxer “.fasta” a la molècula tenim l’opció de pujar el fitxer des de local.

5.5.2 Tractament de Gens

Per la visualització dels gens disposem del menú corresponent a *FormMarca* que es fa visible al seleccionar l'apartat *Gens* del menú. Ens apareix el "Menú Gen" que fa les funcions de crear un nou gen, modificar un existent, esborrar-lo o visualitzar-lo.

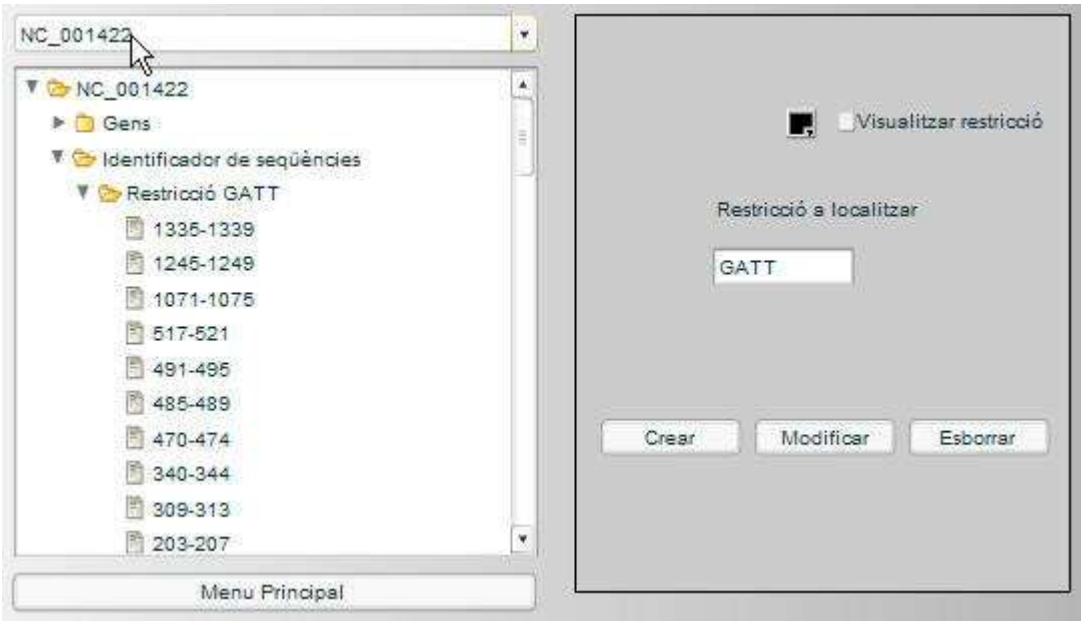


Al menú podem visualitzar els paràmetres del gen i tenim l'opció de fer-lo visible.

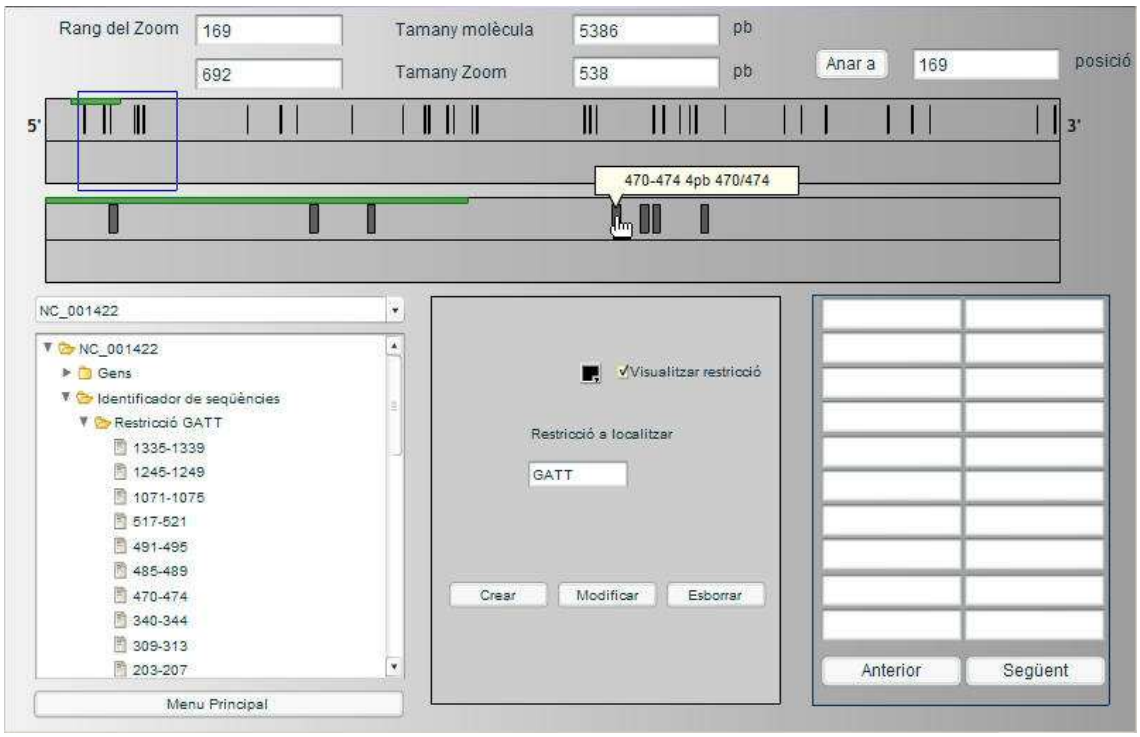


5.5.3 Tractament de seqüències

Per visualitzar les opcions de localització de seqüències seleccionarem “Identificador de seqüències” al menú. Es visualitzarà el menú corresponent a *“FormRestricció”*.

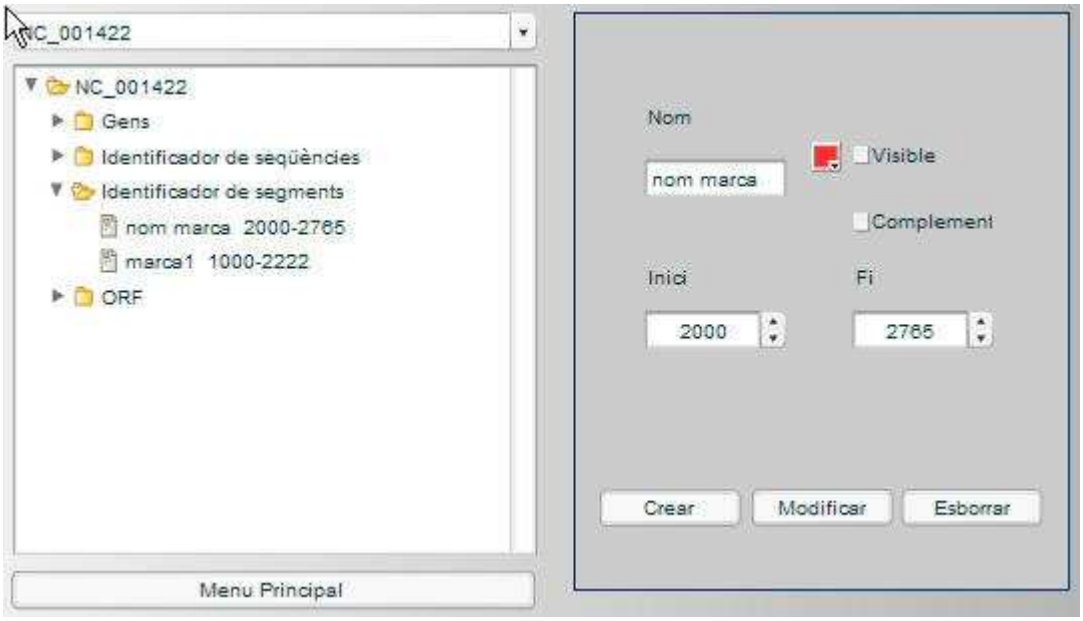


Ens apareixen les opcions de crear, modificar, esborrar i visualització de les seqüències de codi que volem localitzar. L’entrada de seqüències esta limitada a les 4 lletres presents a l’ADN.

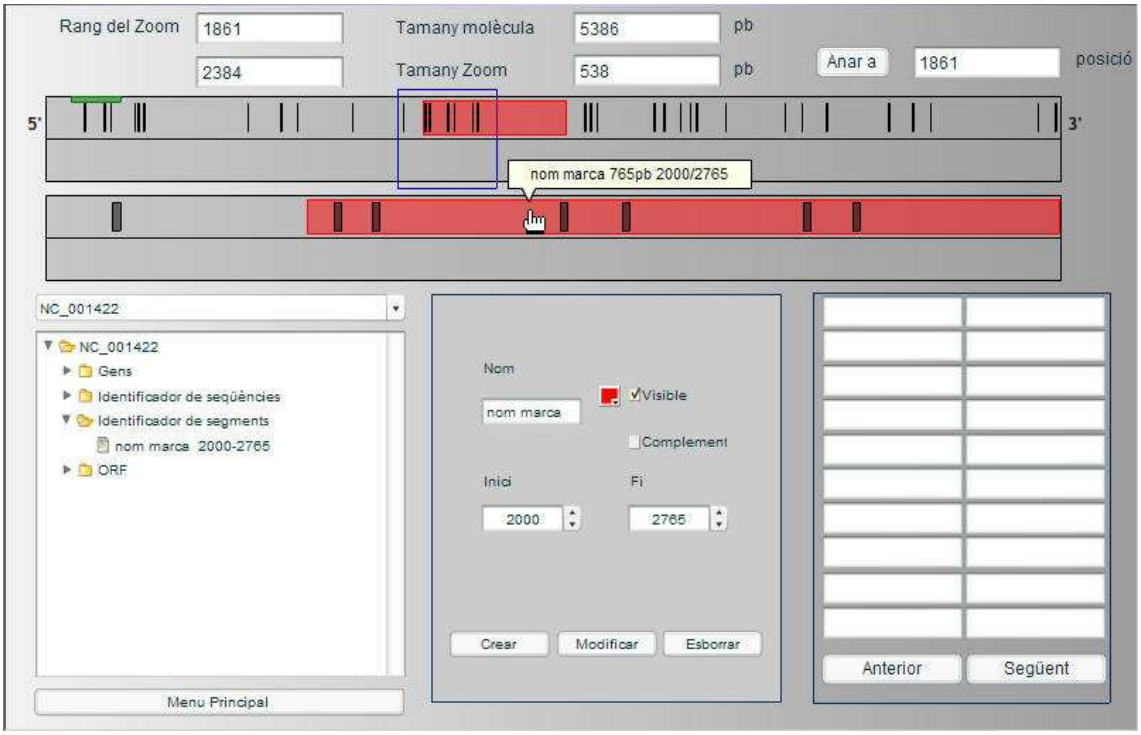


5.5.4 Tractament de Marques

Per visualitzar les opcions d'identificador de zones seleccionarem "Identificador de segments" al menú. Es visualitzarà el menú corresponent a *"FormMarca".*



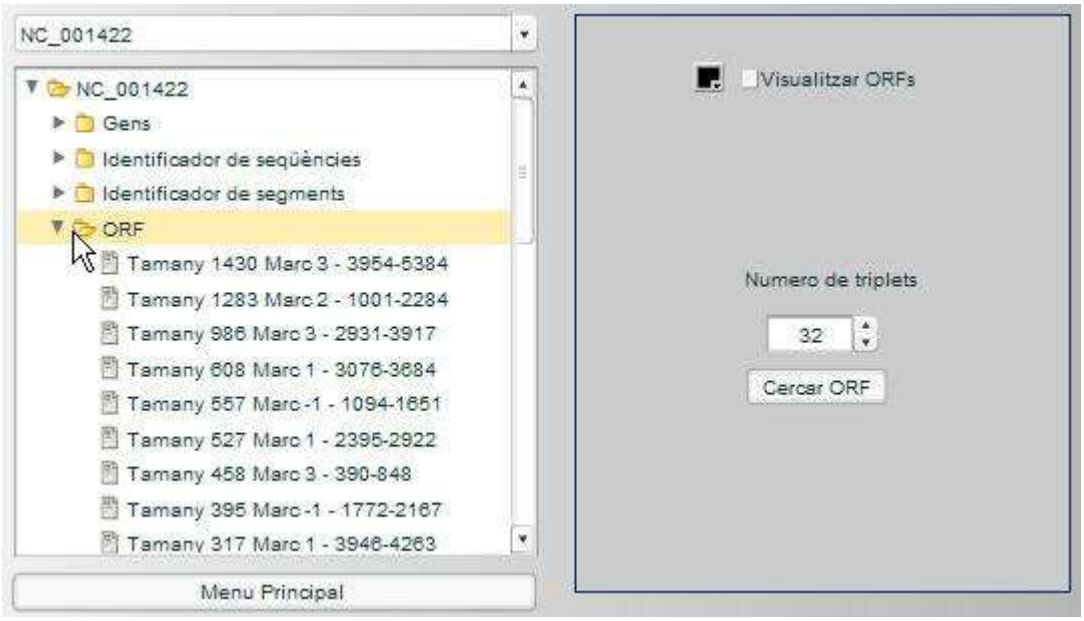
Ens apareixen les opcions de crear, modificar, esborrar i visualització dels segments.



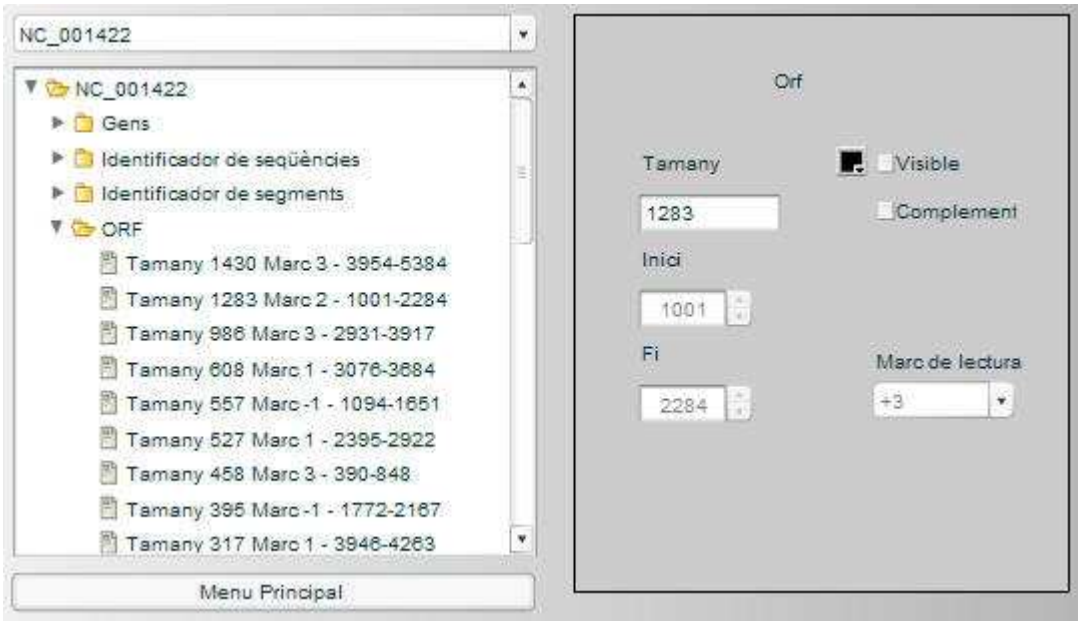
5.5.5 Tractament d'ORF

Els ORF són una mica diferents de la resta, el seu tractament és automàtic. Per visualitzar les opcions d' ORFs seleccionarem "ORF" al menú. Es visualitzarà el menú corresponent a *"FormORFMenu".*

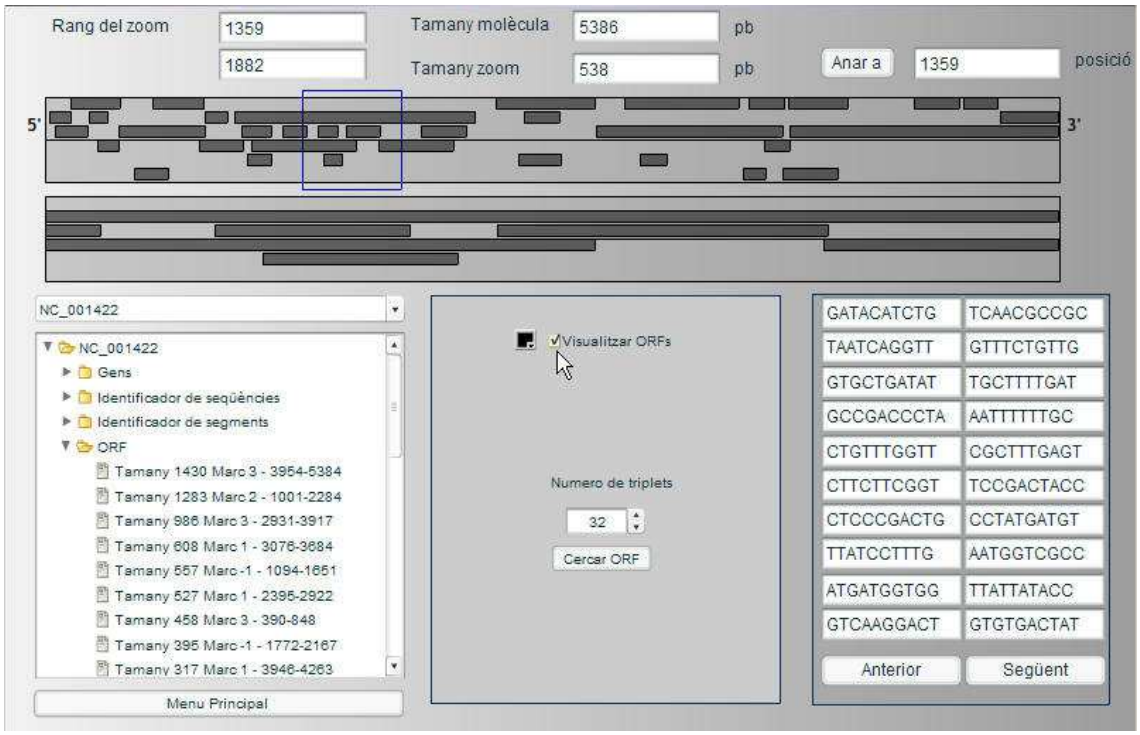
.



Aquest menú ens presenta l'opció de cercar els ORF de la molècula o visualitzar tots els trobats. Si els volem visualitzar per separat només cal seleccionar el que vulguem per que ens aparegui un nou menú de visualització *"FormORF".*



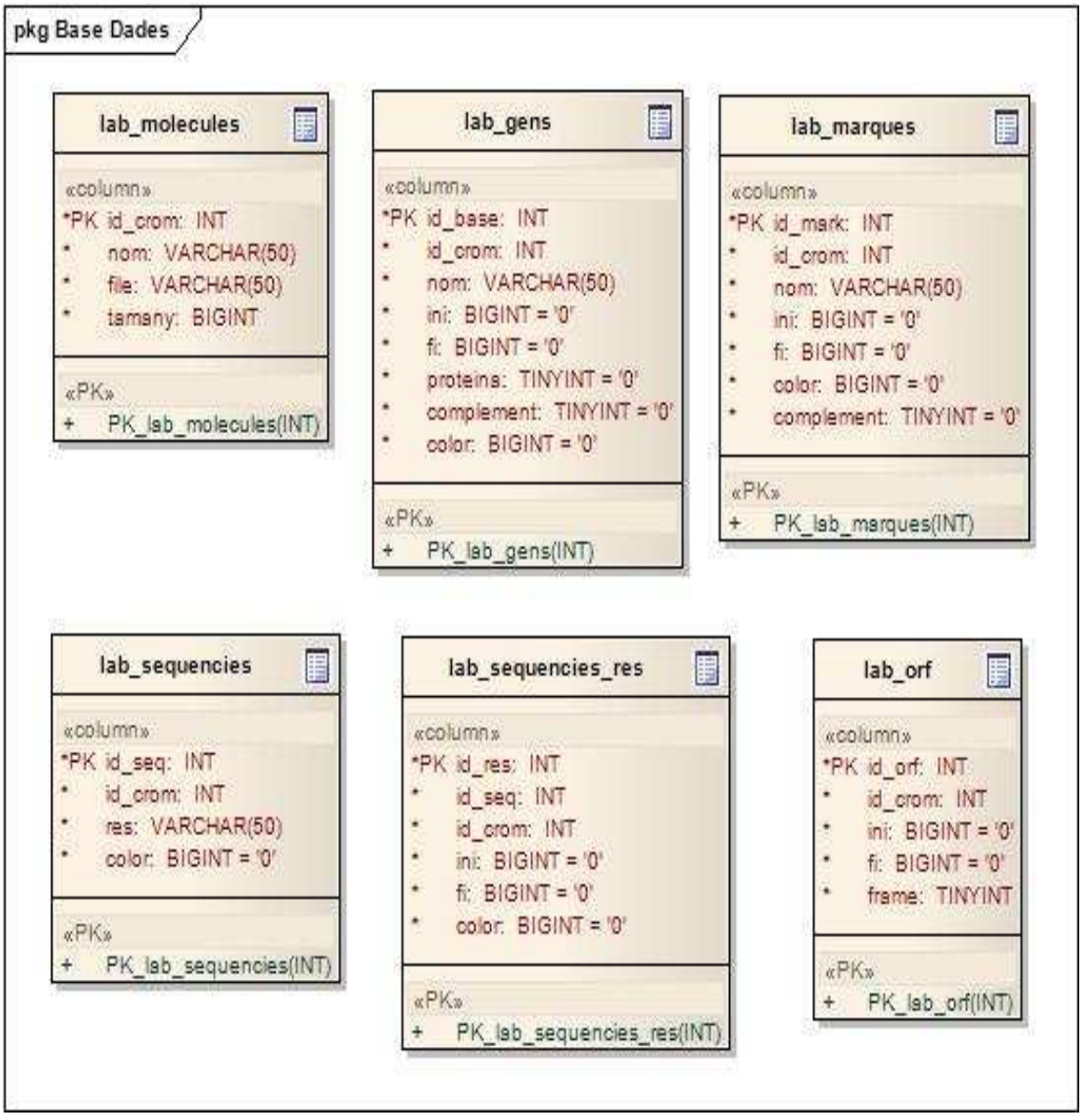
La seva visualització



5.6 Disseny de la Base de Dades

Seguint el model de dades realitzat a l'anàlisi, es crea una estructura de taules en un servidor MYSQL. Per fer-ho, s'ha fet una traducció directa de les classes que apareixen al diagrama cap a taules reals de la base de dades. S'han agregat una sèrie de camps identificadors a les diferents taules per facilitar l' indexació i la seva posterior consulta. El servidor MYSQL també s'encarregarà de verificar que les dades que entrem siguin correctes, evitant així, inconsistències a les taules. Totes les relacions de dependència entre taules es tractaran per codi en PHP per evitar problemes amb les relacions d'algunes versions i tipus de taules en MYSQL.

Un cop feta la traducció tenim les següents taules:



Taula lab_molecules

Guarda la informació de les molècules amb la que treballarem

NOM DEL CAMP	TIPUS DE DADES	Descripció
Id_crom (PK)	Integer AutoIncremental	Identificador de molècula
Nom	Varchar(50)	Nom de molècula
File	Varchar(50)	Nom del fitxer associat a la molècula
Tamany	BigInteger	Numero de parells de bases de la molècula

Taula lab_gens

Guarda la informació dels gens

NOM DEL CAMP	TIPUS DE DADES	Descripció
Id_base (PK)	Integer Autoincremental	Identificador de Gen
Id_crom	Integer	Identificador de molècula
Nom	Varchar(50)	Nom del gen
Ini	Integer	Posició d'inici del gen
Fi	Integer	Posició final del gen
Proteïna	TinyInt(0)	Indica si el gen produeix una proteïna
Complement	TinyInt(0)	Indica si el gen es troba a la cadena simple complementaria
Color	BigInteger	Guarda el color amb el que el volem representar

Taula lab_marques

Guarda la informació de segments que volem visualitzar

NOM DEL CAMP	TIPUS DE DADES	Descripció
Id_mark (PK)	Integer Autoincremental	Identificador del segment
Id_crom	Integer	Identificador de molècula
Nom	Varchar(50)	Nom del segment
Ini	Integer	Posició d'inici del segment
Fi	Integer	Posició final del segment
Complement	TinyInt(0)	Indica si la marca la volem a la cadena simple complementaria
Color	BigInteger	Guarda el color amb el que el volem representar

Taula lab_seqüències

Guarda la informació de les seqüències

NOM DEL CAMP	TIPUS DE DADES	Descripció
Id_seq (PK)	Integer Autoincremental	Identificador de seqüència
Id_crom	Integer	Identificador de molècula
Res	Varchar(50)	Seqüència de lletres a localitzar
Color	BigInteger	Guarda el color amb el que el volem representar

Taula lab_seqüències_res

Guarda la informació de la localització de les seqüències trobades a les molècules

NOM DEL CAMP	TIPUS DE DADES	Descripció
Id_res (PK)	Integer Autoincremental	Identificador de seqüències localitzades
Id_seq	Integer	Identificador de seqüència
Id_crom	Integer	Identificador de molècula
Ini	Integer	Posició d'inici de la seqüència
Fi	Integer	Posició final de la seqüència
Color	BigInteger	Guarda el color amb el que el volem representar

Taula lab_orf

Guarda la informació dels ORF trobats

NOM DEL CAMP	TIPUS DE DADES	Descripció
Id_orf (PK)	Integer Autoincremental	Identificador de l'ORF trobat
Id_crom	Integer	Identificador de molècula
Ini	Integer	Posició d'inici del ORF
Fi	Integer	Posició final del ORF
Color	BigInteger	Guarda el color amb el que el volem representar

5.7 Disseny de fitxers auxiliars

5.7.1 XML

XML, de l'anglès *e**X**tensible **M**arkup **L**anguage* («llenguatge de marques extensible»), és un metallenguatge extensible, d'etiquetes. XML no ha nascut només per a la seva aplicació a Internet, sinó que es proposa com a un estàndard per a l'intercanvi d'informació estructurada entre diferents plataformes. Es pot utilitzar per a bases de dades, editors de text, fulls de càlcul i per moltes altres aplicacions. XML és una tecnologia relativament senzilla. A l'actualitat té un paper molt important, ja que permet la compatibilitat entre sistemes, permetent compartir informació d'una manera segura, fiable i fàcil.

Veiem un exemple de fitxer XML utilitzat a l'aplicació.

```
<COMBOBOX label="etiqueta">
        <CROM nom="NC_001422" id="1"/>
</COMBOBOX>
```

Aquest exemple pertany al fitxer que carrega el component ComboBox (el desplegable) de l'aplicació que permet seleccionar una molècula del llistat de molècules entrades.
En aquest exemple tenim una etiqueta general anomenada COMBOBOX amb un atribut anomenat label amb el valor "etiqueta". Aquesta etiqueta general té una etiqueta filla CROM amb atributs nom i id. Aquí s'aniran afegint noves etiquetes CROM filles de COMBOBOX per cada molècula que afegim al sistema.

A continuació un exemple d'obtenció de l' informació del XML

```
Var xmlTree = new XML();
Var nom = String;

xmlTree.ignoreWhite = true;
xmlCombo.load('Combo.xml');
nom=xmlCombo.firstChild.childNodes[0].attributes.nom;
```

Amb aquest petit exemple de codi ja hauríem agafat el valor de l'atribut nom del primer fill del fitxer XML.

L'aplicació treballa amb dos fitxers XML. El fitxer de selecció de molècules, el de l'exemple que hem vist i el que guarda l' informació de una molècula.
Recordem que aquests fitxers es generen a partir de la Base de Dades i que s'utilitzen per carregar l' informació en diversos components de Flash.

Exemple reduït del fitxer xml de la molècula NC_001422.

```
<MENU label="NC_001422" menu="0" id="1" file="NC_001422.fasta" tamany="5386">
        <BASES label = "Gens" menu="1">
                <GEN label = "phiX174p01  3981-136  Proteïna" id="1" nom="phiX174p01" ini="3981" fi="136" nivell=""
                proteina="1" complement="0" color="0" menu="100"/>
                <GEN label = "phiX174p02  4497-136  Proteïna" id="2" nom="phiX174p02" ini="4497" fi="136" nivell=""
                proteina="1" complement="0" color="100" menu="100"/>
        </BASES>
        <SEQUENCIES label = "Identificador de seqüències" menu="2">
                <RESTRICCIO label = "Restricció GATT" id="1" res="GATT" menu="200">
                        <SEQ label = "1335-1339" id_seq="1" id_res="20" id_crom="1" ini="1335" fi="1339" color="0"
                        menu="2000"/>
                        <SEQ label = "1245-1249" id_seq="1" id_res="19" id_crom="1" ini="1245" fi="1249" color="0"
                        menu="2000"/>
                </RESTRICCIO>
        </SEQUENCIES>
        <MARQUES label = "Identificador de segments" menu="4">
                <MARCA label = "marca1  1000-2222" id="2" nom="marca1" ini="1000" fi="2222" color="0"
                complement="0" menu="400"/>
        </MARQUES>
        <ORF label = "ORF" menu="5">
                <ORF label = "Tamany 275 Marc 1 - 568-843" id="37" frame="1" ini="568" fi="843" menu="500"/>
                <ORF label = "Tamany 260 Marc 1 - 133-393" id="36" frame="1" ini="133" fi="393" menu="500"/>
        </ORF>
</MENU>
```

5.7.2 FASTA

El format FASTA és un format de fitxer de text àmpliament utilitzat en bioinformàtica, utilitzat per representar seqüències de nucleòtids on els parells de bases o aminoàcids es representen utilitzant els codi d'una sola lletra. El format també permet incloure noms de seqüències i comentaris.

El format simple FASTA, fa que sigui fàcil de treballar i analitzar en llenguatges com PERL i Phyton
Una seqüència en format fasta comença amb una descripció en una sola línia (capçalera) seguida de la seqüència de dades. La capçalera es diferencia de la resta per començar amb el símbol '>'. La següent paraula és l'identificador de la seqüència i la resta la descripció. La seqüència finalitza si aparegués un altre identificador de capçalera.

Veiem un exemple:

```
>gi|17981852:1-16571 Homo sapiens mitochondrion, complete genome
GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGCTCTCCATGCATTTGGTATTTTCGTCTGGGGG
GTGTGCACGCGATAGCATTGCGAGACGCTGGAGCCGGAGCACCCTATGTCGCAGTATCTGTCTTTGATTC
CTGCCTCATTCTATTATTTATCGCACCTACGTTCAATATTACAGGCGAACATACCTACTAAAGTGTGTTA
```

Cal remarcar que l'aplicació treballa amb una variant del format sense capçaleres i amb línies de 70 caràcters.

5.8 Implementació

5.8.1 Cerca i entrada dels ORF a la Base de Dades

Donat un identificador de molècula “id_crom” i el número de triplets mínims, eliminem els antics ORF calculats per aquesta molècula i llancem el procés de cerca dels nous ORF. Un cop finalitzada la cerca es captura la sortida i s’afegeix a la BD.

(arbre_xml.php, línia 376)

```
function new_orf($id_crom,$triplets){
        $BD = new DB();
        $query= "delete from ".BDORF." where id_crom='$id_crom'";
        $QCROM=$BD->query($query);
        $query="select * from ".BDCROM." where id_crom=".$id_crom;
        $QCROM=$BD->query($query);
        $CROM = mysql_fetch_assoc($QCROM);
        $BD->free($QCROM);
        $BD->close();

        $nom=escapeshellcmd($nom);
        $cmd='perl "'.CONF_PATH.'orfs.pl" "'.MOL_PATH.$CROM['nom'].'.fasta" '.$triplets;

        $cali=exec($cmd,$sortida);
        print_r($out);

        $BD = new DB();
        foreach ($sortida as $aux) {
                list($ini,$fi,$frame,$tamany)=split(":",$aux);
                $query= "insert into ".BDORF." (id_crom,ini,fi,frame) values ($id_crom,
                $ini, $fi, $frame)";
                $QORF=$BD->query($query);
        }
        $BD->close();
        return $cali." ".$cmd." ".$sortida[1];
}
```

5.8.2 Algoritme ORF

Un ORF des del punt de vista informàtic és la localització de patrons de 3 caràcters que ens indiquen l’inici del ORF i 3 caràcters més que indiquen el final del ORF. Aquesta cerca s’ha de realitzar amb 3 offsets diferents d’inici que indiquen el marc de lectura i 3 més per la cadena inversa. La cadena inversa és una traducció de les lletres del ADN (A – T i G – C).

Tenim un fitxer en format text amb línies formades per seqüències dels caràcters ‘A’,’C’,’G’,’T’. Partint de 3 posicions inicials [0,1,2] hem de cercar uns patrons d’inici anomenat codó inicial, format pels caràcters (“ATG”), i un dels codons finalitzadors format per una de les 3 seqüències finalitzadores ("TAA”, "TGA”, "TAG"). La seqüència de caràcters entre un codó inicial i un final és un ORF.

(orfs.pl)

```
#! /usr/bin/perl -w
use integer;
use strict;

my $usage = "orfFind.pl [genome in FASTA format] [min length of ORF]\n";
my $line="";
my $ch="";
my $genome="";
my @EcoliORFs='';

die $usage unless @ARGV == 2;

open (SEQ, "<$ARGV[0]") or die "Cannot open $ARGV[0]:$!\n";

while ($line = <SEQ>) {
    $ch=uc(chop($line));
    if ($ch=~m/A|T|G|C/) { $line.=$ch; }
    $genome .= uc($line) unless $line =~ m/^>/;
}

for my $j ( 0 .. 2 ) {
  push (@EcoliORFs, getORF($genome,$j,$ARGV[1],0));
}

$genome=~tr/AGTC/TCAG/;
my $complementaria=reverse($genome);

#print $complementaria."\n";

for my $j ( 0 .. 2 ) {
  push (@EcoliORFs, getORF($complementaria,$j,$ARGV[1],1));
}

foreach my $EcoliORF (@EcoliORFs) {
  print "$EcoliORF\n";
}

sub getORF {
  my $dna      = $_[0];    #DNA
  my $offset   = $_[1];       #Frame / Marc de lectura
  my $len      = $_[2]*3;     #Length  / Tamany
  my $comp     = $_[3];       #Complementari / Cadena normal o la complementaria
  my $i        = 0;
  my $seq      = "";
  my @ORFs     = ();
  my $dif              = 0;
  my $start_bit = 0;
  my $ini      = 0;
  my $codon            = "";
  my $start            = 0;
  my $end              = 0;
  my $frame            = $offset;
  my $tamany   = length($dna);
  substr($dna,0,$offset,"");

  while (length($dna) > 0 ) {
    $codon = substr($dna, 0, 3, "");
    if ($codon eq "ATG") {
        if ($start_bit == 0){
                $start = ($i * 3) + $offset;
                $start_bit=1;
        }
    } elsif (((($codon eq "TAA") or ($codon eq "TGA") or ($codon eq "TAG") or
(length($dna)<3)) and $start_bit==1) {
                $end = ($i * 3) + 3 + $offset;
                $seq .= $codon;
                $start++;
#               print $seq."\n";

                if ( $start > $end ) {
                        $dif = $tamany+$end-$start;
                } else {
                        $dif = $end - $start;
                }
```

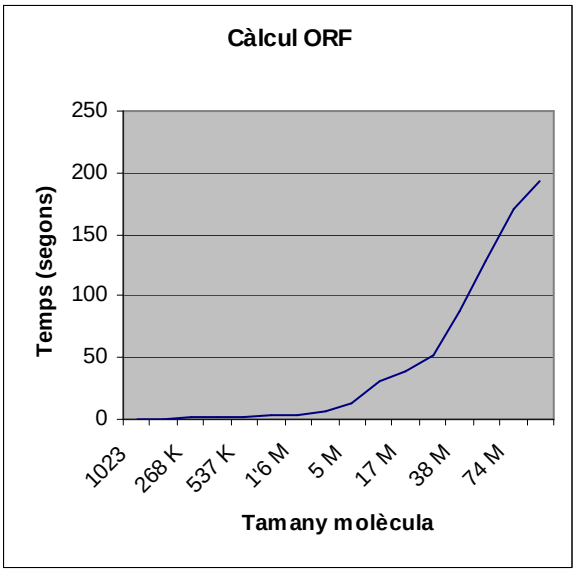
```
                $dif++;
                $frame= $offset;
                if ($comp == 1 ) {
                        my $aux=$end;
                        $end=$tamany-$start+1;
                        $start=$tamany-$aux+1;
                        if ($frame==0) {$frame=-1;}
                        elsif ($frame==1) {$frame=-2;}
                        elsif ($frame==2) {$frame=-3;}
                } else {
                        if ($frame==0) {$frame=1;}
                        elsif ($frame==1) {$frame=2;}
                        elsif ($frame==2) {$frame=3;}
                }

                if ($start_bit == 1 and length($seq) >= $len) {

                        push (@ORFs, "$start:$end:$frame:$dif");
                }
                $seq = "";
                $start_bit = 0;
    }
    $seq .= $codon if $start_bit == 1;
    ++$i;
  }
  return @ORFs;
}
```

Taula de temps del programa orfs.pl

Parell de bases	Temps (segons)
1023	0
134 K	0
268 K	1
398 K	1
537 K	1
1 M	3
1'6 M	4
3 M	6
5 M	13
13 M	31
17 M	39
23 M	52
38 M	88
56 M	128
74 M	171
84 M	193



5.8.3 Generació de fitxers XML

Hi han dos fitxers XML generats amb les dades de la Base de Dades, el llistat de molècules i el fitxer de cada molècula.

Veurem la generació del XML associat a una molècula determinada. Li passem com a paràmetres l'ID de la molècula creada el nom, fitxer i tamany. Amb aquestes dades anem realitzant una sèrie de consultes a la Base de Dades de les diferents parts que pot tenir

associada una molècula. Guardant aquestes dades al fitxer que porta com a nom el nom de la molècula consultada.

(arbre_xml.php, linea 69)

```
function genera_arbre_id($id,$nom,$file,$tamany) {
	$XMLfile=fopen(XML_PATH.$nom.'.xml','w+');
	$dades = '< ?xml version="1.0" encoding="iso-8859-1"?>'."\n";
	$dades	=	'<MENU	label="'.$nom.'"	menu="0"	id="'.$id.'"
file="'.$file.'"tamany="'.$tamany.'">'."\n";
	fwrite($XMLfile,$dades);
	$BD = new DB();
	$this->genera_bases($id,$XMLfile,$BD);
	$this->genera_seq($id,$XMLfile,$BD);
	//$this->genera_exons($id,$XMLfile,$BD);
	$this->genera_marques($id,$XMLfile,$BD);
	$this->genera_orf($id,$XMLfile,$BD,$tamany);
	$BD->close();
	$dades = '</MENU>'."\n";
	fwrite($XMLfile,$dades);
	fclose($XMLfile);
}
```

Entrem a veure com es genera per exemple la part de seqüències amb la funció genera_seq($id,$XMLFile,$BD).

(arbre_xml.php, linea 135)

```
function genera_seq($id,$file,$BD){
	$dades='	<SEQUENCIES label = "Identificador de seqüències" menu="2">'."\n";
	fwrite($file,$dades);
	$query="select * from ".BDSEQ." where id_crom=".$id;
	$QSEQ=$BD->query($query);

	if (mysql_num_rows($QSEQ)>0) {
		$row_QSEQ = mysql_fetch_assoc($QSEQ);

		do {
			$Llista_SEQ[]=array(
				"id_seq"=>$row_QSEQ['id_seq'],
				"res"=>$row_QSEQ['res']
			);
		} while ($row_QSEQ = mysql_fetch_assoc($QSEQ));

		$BD->free($QSEQ);

		foreach ($Llista_SEQ as $SEQ) {
			$label='Restricció '.$SEQ['res'];
			$dades='<RESTRICCIO label = "'.$label.'" id="'.$SEQ['id_seq'].'"
res="'.$SEQ['res'].'" menu="200">'."\n";
			fwrite($file,$dades);
			$query="select * from ".BDRES." where id_seq=".$SEQ['id_seq'];
			$QSEQ=$BD->query($query);
			if (mysql_num_rows($QSEQ)>0) {
				$row_SEQ = mysql_fetch_assoc($QSEQ);
				unset($Llista_SEQ);
				do {
					$Llista_SEQ[]=array(
						"id_seq"=>$row_SEQ['id_seq'],
						"id_res"=>$row_SEQ['id_res'],
						"id_crom"=>$row_SEQ['id_crom'],
						"ini"=>$row_SEQ['ini'],
						"fi"=>$row_SEQ['fi'],
						"color"=>$row_SEQ['color']
					);
				} while ($row_SEQ = mysql_fetch_assoc($QSEQ));

				$BD->free($QSEQ);

				foreach ($Llista_SEQ as $SEQ) {
					$label=$SEQ['ini']."-".$SEQ['fi'];
```

```
                                        $dades='<SEQ       label       =       "'.$label.'"
id_seq="'.$SEQ['id_seq'].'"   id_res="'.$SEQ['id_res'].'"   id_crom="'.$SEQ['id_crom'].'"
ini="'.$SEQ['ini'].'" fi="'.$SEQ['fi'].'" color="'.$SEQ['color'].'" menu="2000"/>'."\n";
                                        fwrite($file,$dades);
                                }
                        }
                        $dades='       </RESTRICCIO>'."\n";
                        fwrite($file,$dades);
                }
        }
        $dades='       </SEQUENCIES>'."\n";
        fwrite($file,$dades);
}
```

5.8.4 Consultes a la Base de Dades

Eliminació de Molecula

Al disseny hem comentat que el tractament d'integritat referencial de la Base de Dades es feia per codi, ara veurem com s'elimina una molècula i tot el contingut relacionat amb ella.

(arbre_xml.php linia 497)

```
function del_molecula($id_crom) {
        $BD = new DB();
        $query= "delete from ".BDBASES." where id_crom='$id_crom'";
        $Queri=$BD->query($query);
        $query= "delete from ".BDSEQ." where id_crom='$id_crom'";
        $Queri=$BD->query($query);
        $query= "delete from ".BDMARKS." where id_crom='$id_crom'";
        $Queri=$BD->query($query);
        $query= "delete from ".BDEXONS." where id_crom='$id_crom'";
        $Queri=$BD->query($query);
        $query= "delete from ".BDCDS." where id_crom='$id_crom'";
        $Queri=$BD->query($query);
        $query= "delete from ".BDORF." where id_crom='$id_crom'";
        $Queri=$BD->query($query);
        $query= "delete from ".BDRES." where id_crom='$id_crom'";
        $Queri=$BD->query($query);
        $query= "delete from ".BDCROM." where id_crom='$id_crom'";
        $Queri=$BD->query($query);
        $BD->close();
}
```

Creació d'una sequencia

Recordem que una seqüència és un conjunt de lletres dels caràcters ("A","G","C","T") que s'han de localitzar al fitxer FASTA de la molècula.

(arbre_xml.php, linia 335)

```
function new_seq($id_crom,$res,$color){
        $BD = new DB();
        $query=      "insert      into      ".BDSEQ."      (id_crom,res,color)      values
('".$id_crom."','".$res."','".$color."')";
        $QSEQ=$BD->query($query);
        $id=mysql_insert_id();
        $BD->close();
        $this->new_res($id_crom,$id,$res);
        return "Sequencia entrada $id|".$id_crom."|".$res."|".$color."|fin";
}
```

Un cop creada l'entrada corresponent a la seqüència es busquen les posicions on es localitza per entrar-les a una nova taula.

(arbre_xml.php, linia 408)

```
function new_res($id_crom,$id_cds,$res) {
        $BD = new DB();
        $query="select * from ".BDCROM." where id_crom=".$id_crom;
        $QCROM=$BD->query($query);
        $CROM = mysql_fetch_assoc($QCROM);
        $BD->free($QCROM);
        $query= "delete from ".BDRES." where id_seq='$id_cds'";
        $Queri=$BD->query($query);
        $BD->close();
        $nom=escapeshellcmd($nom);
        $cmd='perl "'.CONF_PATH.'sequencia.pl" "'.MOL_PATH.$CROM['nom'].'.fasta" '.$res;
        $cali=exec($cmd,$sortida);
        $BD = new DB();
        foreach ($sortida as $aux) {
                list($ini,$fi)=split(":",$aux);
                $query=    "insert    into    ".BDRES."    (id_crom,id_seq,ini,fi)    values
        ($id_crom,$id_cds,$ini,$fi)";
                $QRES=$BD->query($query);
        }
        $BD->close();
        return "new_res";
}
```

Modificació d'un gen

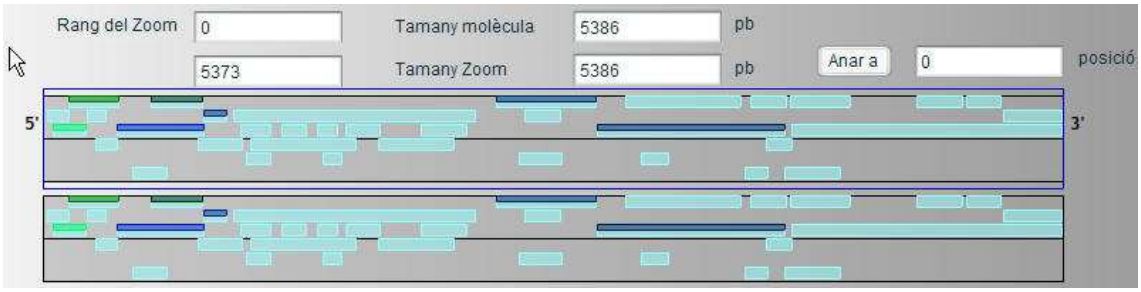
(arbre_xml.php, linia 447)

```
function mod_base($id_base,$id_crom,$nom,$ini,$fi,$proteina,$complement,$color){
        $BD = new DB();
        $query=  "update  ".BDBASES."  set  id_crom='$id_crom',  nom='$nom',  ini='$ini',
fi='$fi',    proteina='$proteina',    complement='$complement',    color='$color'    where
id_base='$id_base'";
        $QBASES=$BD->query($query);
        $BD->close();
}
```

5.8.5 Visualització i Zoom

La part més important de l'aplicació és la representació dels elements i el zoom realitzat sobre ells.

Per una part tenim els "elements" que visualitzem; que es representen com a rectangles d'un tamany i alçada determinats, pel tipus d'element que representa i les seves dimensions relatives al tamany de pantalla o tamany de molècula. Entenem com a element cada rectangle visualitzar.



Amb el tamany total de la molècula i coneixent el tamany de la pantalla és fàcil calcular el tamany que ha de simbolitzar. Ens faltarà calculat la posició inicial respecte el total de la molècula.

```
vistax=Number(vista._width)-2.5;
posIni=((vistax*posIni)/(Number(tamany)-2));
posFi=((vistax*fi)/(Number(tamany)-2));
Ancho=posFi-posIni;.
```

Veiem que és necessari introduir petites correccions a les formules per motius de visualització i els propis pixels dels marges del dibuix.

Un cop tenim la posició inicial, final, alçada, amplada i color creem dinàmicament l'objecte que representarà l'element que volem visualitzar. Passant-li paràmetres com el nom, posició inicial i final reals en la molècula per utilitzar-los posteriorment.

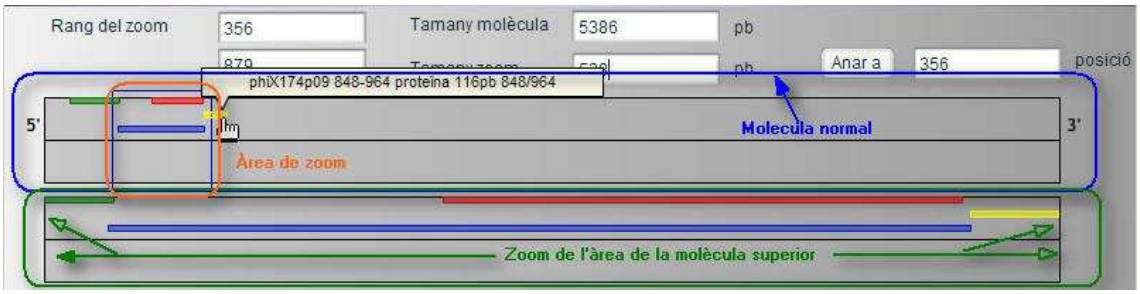
```
var thisBug:MovieClip=this.attachMovie("Element",idBug+"_mc",
this.getNextHighestDepth(), {_seguent:continua,_nom:nom, _x:posIni,
_y:posY, _color:color, _width:Ancho, _height:Alto, _infoposIni:posini,
_infoposFi:posfi, _POSINI:ini, _POSFI:fi, _tamany:tamany});
```

I el seu element zoom.

```
var thisBugZoom:MovieClip = this.attachMovie("Element",idBug+"_Zoom",
this.getNextHighestDepth(), {_seguent:continua,_nom:nom, _x:posIni,
_y:posY+70, _color:color, _width:Ancho, _height:Alto, _infoposIni:posini,
_infoposFi:posfi, _POSINI:ini, _POSFI:fi, _tamany:tamany});
```

Tot element creat s'afegeix a una taula per facilitar la seva cerca i eliminació.

Per fer el zoom utilitzem un requadre que simbolitza l'ampliació de la zona que estem representant a l'àrea de zoom i el tamany entrat en la casella "Tamany zoom".



El requadre o zoom el movem clicant sobre la molècula. Així obtenim la posició on s'ha clicat per moure el requadre i iniciar el càlcul dels elements que s'estan mostrant dins del requadre de zoom. Aquesta posició també s'utilitza per demanar el codi de seqüència

corresponent a aquella posició i carregar-lo a l'objecte *"Codi"* per poder visualitzar-lo i manipular-lo.

```
function VistaPress():Void{
      _parent.dispatchEvent ( {type: "obtenirCodi" } );
      _parent.Moure_Marc();
      _parent.Posicio();
      _parent.FerZoom();
}
```

Per fer el zoom calculem el rang inicial i final del requadre zoom i el contrastem amb tots els elements que s'estan visualitzant per veure si estan dins del rang. Aquí utilitzem la taula d'elements visibles on afegim cada element que es mostra i els paràmetres que passem a cada element quan el creem.

```
var cuadre:Object = {ini:0,fi:0};
this.Marc_pos(cuadre);

for (var i=0;i<Visibles.length;i++){
       Buscar_elements(String(Visibles[i]),cuadre,xzoom);
}
```

Per cada element que trobem dins del rang es calculen els paràmetres d'inici, final i tamany en funció del tamany del zoom per representar-los. Els elements fora del rang s'eliminen del zoom.

```
if (this[elem+"_mc"].Visible()) {
      ca=this[elem+"_Zoom"].dins_marc(cuadre.ini,cuadre.fi);
      if (ca>-1) {
            Modificar_element(elem+"_Zoom",cuadre,xzoom);
            this[elem+"_Zoom"].Show();
      } else {
            this[elem+"_Zoom"].Hide();
      }
      seguent=this[elem+"_mc"].seguent();
      if (seguent!= null) {
            Buscar_elements(elem+'S',cuadre,xzoom);
      }
}
```

Hi ha elements que tenen un element "següent" associat. Es troba en molècules circulars que poden tenir un element (generalment gens) situat entre el final i l'inici de la representació. Aquest elements s'han de contemplar ja que es representen per separat però són el mateix gen.

(FormMostrar.as)

```
import com.laboratori.view.zonatreball.Visor.Element;
import mx.controls.*;
import com.ariaware.arp.ArpForm;
import mx.utils.Delegate;

class com.laboratori.view.zonatreball.Visor.FormMostrar extends ArpForm {
       var vista:MovieClip;
```

```
        var vistaZoom:MovieClip;
        var vistaCodi,lupa:MovieClip;
        var marc:MovieClip;
        var EditTamany:TextInput;
        var EditZoom:TextInput;
        var EditPosicio:TextInput;
        var MarcMin:TextInput;
        var MarcMax:TextInput;
        var BtAnar:Button;
        var zoom:Number=1;
        var MAX:Number=100;
        var Visibles=Array();
        var VisiblesID=Array();

        var bugNum,bugNumZoom:Number=0;

        function FormMostrar() {
                trace("formMostrar ");
        }

        function onLoad() {
                Visibles=new Array();
                vista.onMouseMove=VistaMouseMove;
                vista.onPress=VistaPress;
                BtAnar.addEventListener ("click",Delegate.create(this, BtAnarClick));
        }

        function Posicio():Void{
                var total,vistax,x,posicio:Number;
                vistax=vista._width;
                total=Number(EditTamany.text);
                x=marc._x;
                posicio=Math.round((total*x)/vistax);
                EditPosicio.text=String(posicio);
        }

        function VistaPress():Void{
                _parent.dispatchEvent ( {type: "obtenirCodi" } );
                _parent.Moure_Marc();
                _parent.Posicio();
                _parent.FerZoom();
        }

        function VistaMouseMove(){
                _parent.lupa._x = this._xmouse-9;
        _parent.lupa._y = this._ymouse-9;
        }

        function randRange(min:Number, max:Number):Number {
                var randNum:Number = Math.round(Math.random()*(max-min))+min;
                return randNum;
        }

        function
addBug(idBug:String,nom:String,posIni:Number,posFi:Number,Nivell:Number,Alto:Number,
color:Number, complement:Number):Void {
                var posY,Ancho:Number=0;
                var continua:String=null;
                var posfi:Number=posFi;
                var posini:Number=posIni;
                var ini:Number=posIni;
                var fi:Number=posFi;
                var tamany,vistax,minibug,pb:Number;

                tamany=Number(EditTamany.text);

                if (Number(posFi)<Number(posIni)) {
                        minibug=posFi;
                        fi=tamany;
                        continua=idBug+'S';
                        addBug(continua,nom,0,minibug,Nivell,Alto,color,complement);

                } else {
                        continua=null;
                }

                vistax=Number(vista._width)-2.5;
```

```
		switch (Nivell){
			case 0:
				posY=30;//10;
			break;
			case 1:
				posY=10;//30;
			break;
			case 2:
				posY=20;//20;
			break;
			case 4:
				posY=27;
			break;
			case 5:
				posY=25;
			break;
		}

		if (complement==0) {
			switch (Nivell){
				case 0:
					posY=-10;//10;
				break;
				case 1:
					posY=-30;//30;
				break;
				case 2:
					posY=-20;//20;
				break;
				case 4:
					posY=-27;
				break;
				case 5:
					posY=-25;
				break;
			}
		} else {
			if (Nivell==4){
				posY-=24;
			} else {
				posY-=10;
			}
		}
		posIni=((vistax*posIni)/(Number(tamany)-2));
		posFi=((vistax*fi)/(Number(tamany)-2));
		Ancho=posFi-posIni;

		var  thisBug:MovieClip          =  this.attachMovie("Element",idBug+"_mc",
this.getNextHighestDepth(),     {_seguent:continua,_nom:nom,     _x:posIni,     _y:posY,
_color:color,    _width:Ancho,    _height:Alto,    _infoposIni:posini,    _infoposFi:posfi,
_POSINI:ini, _POSFI:fi, _tamany:tamany});
		var   thisBugZoom:MovieClip   =   this.attachMovie("Element",idBug+"_Zoom",
this.getNextHighestDepth(),     {_seguent:continua,_nom:nom,     _x:posIni,     _y:posY+70,
_color:color,    _width:Ancho,    _height:Alto,    _infoposIni:posini,    _infoposFi:posfi,
_POSINI:ini, _POSFI:fi, _tamany:tamany});

		Visibles.push(idBug);
		VisiblesID.push(idBug+"_mc");
		bugNum++;
		bugNumZoom++;
	}

	function setTamany(t:Number):Void{
		EditTamany.text=String(t);
		EditZoom.text=EditTamany.text;
		EditPosicio.text=String(0);
	}

	function getTamany(Void):Number{
		return Number(EditTamany.text);
	}

	function Show() {
		this._visible = true;
	}
```

```
function Hide() {
	this._visible = false;
}

function BtAnarClick() {
	dispatchEvent ( {type: "obtenirCodi" } );
}

function FerZoom() {
	var x:Number;
	var xzoom,zoomIni,zoomFi:Number;
	var tamany,idelement:Number;
	var ini,fi,zoomini,zoomfi:Number;
	var vista,marco,marcx,marc_ini,marc_fi:Number;

	xzoom=Number(EditZoom.text);

	var cuadre:Object = {ini:0,fi:0};
	this.Marc_pos(cuadre);

	for (var i=0;i<Visibles.length;i++){
		Buscar_elements(String(Visibles[i]),cuadre,xzoom);
	}
}

function Buscar_elements(elem:String,cuadre:Object,xzoom:Number){
	var seguent:String;
	var caca:Number;
	if (this[elem+"_mc"].Visible()) {
		ca=this[elem+"_Zoom"].dins_marc(cuadre.ini,cuadre.fi);
		if (ca>-1) {
			Modificar_element(elem+"_Zoom",cuadre,xzoom);
			this[elem+"_Zoom"].Show();
		} else {
			this[elem+"_Zoom"].Hide();
		}
		seguent=this[elem+"_mc"].seguent();
		if (seguent!= null) {
			Buscar_elements(elem+'S',cuadre,xzoom);
		}
	}
}

function Modificar_element(element:String,marco:Object,xzoom:Number){
	var vistax,elem_ini,elem_fi,mod_ini,mod_fi:Number;
	vistax=vista._width-2.5;

	elem_ini=this[element].posIni();
	elem_fi=this[element].posFi();

	mod_ini=elem_ini-marco.ini;
	mod_fi=elem_fi-marco.ini;

	if (mod_ini<0){
		mod_ini=0;
	}
	if (mod_fi>xzoom){
		mod_fi=xzoom;
	}
	elem_ini=Math.round((vistax*mod_ini)/xzoom);
	elem_fi=Math.round((vistax*mod_fi)/xzoom);

	this[element].X(elem_ini);
	this[element].Ample(elem_fi-elem_ini);
}

function Marc_pos(cuadre:Object){
	var inimarc,fimarc,total,vistax,tzoom,x:Number;
	vistax=vista._width;
	tzoom=Number(EditZoom.text);
	total=Number(EditTamany.text);
	x=this.marc._x;

	inimarc=Math.round((total*x)/vistax);
	fimarc=Math.round((total*(x+this.marc._width))/vistax)-1;

	cuadre.ini=inimarc;
```

```
        cuadre.fi=fimarc;
        MarcMin.text=inimarc;
        MarcMax.text=fimarc;
        return cuadre;
}

function Moure_Marc(){
        var marc,inimarc,fimarc,total,vistax,tamany,x,Xadn:Number;
        vistax=vista._width;
        tamany=Number(EditZoom.text);
        total=Number(EditTamany.text);
        x=this._xmouse;
        marc=Math.round(((tamany*vistax)/total)/2);
        if (marc>x){
                inimarc=0;
        } else if (x >vistax-marc) {
                inimarc=vistax-(marc*2);
        } else {
                inimarc=x-marc;
        }
        fimarc=inimarc+marc*2;
        this.marc._x=inimarc;
        this.marc._width=(marc*2)-1;
}

function Moure_Marc_Pos(pos:Number){
        var marc,inimarc,fimarc,total,vistax,tamany,x,Xadn:Number;
        vistax=vista._width;
        tamany=Number(EditZoom.text);
        total=Number(EditTamany.text);
        x=this._xmouse;
        marc=Math.round(((tamany*vistax)/total)/2);
        if (marc>x){
                inimarc=0;
        } else if (x >vistax-marc) {
                inimarc=vistax-(marc*2);
        } else {
                inimarc=x-marc;
        }
        fimarc=inimarc+marc*2;
        this.marc._x=inimarc;
        this.marc._width=(marc*2)-1;
}

function get_index() {
        return Number(EditPosicio.text);
}

function Seguent(){
        var pos,max:Number;
        max=Number(EditTamany.text);
        pos=Number(EditPosicio.text);
        if (pos > max-200) {
                pos=max-201;
        } else {
                pos=pos+200;
        }
        EditPosicio.text=pos;
        dispatchEvent ( {type: "obtenirCodi" } );
        Moure_Marc();
        FerZoom();
}

function Anterior(){
        var pos:Number;
        pos=Number(EditPosicio.text);
        if (pos < 201) {
                pos=0;
        } else {
                pos=pos-200;
        }
        EditPosicio.text=String(pos);
        dispatchEvent ( {type: "obtenirCodi" } );
        Moure_Marc();
        FerZoom();
}
```

```
    function Destroy(element:String){
        for (var i=0;i<Visibles.length;i++){
                if (String(VisiblesID[i])==String(element)) {
                        Visibles.splice(i,1);
                        VisiblesID.splice(i,1);
                }
        }
        this[element].Destroy();
    }

    function Destroy_all(){
        while (Visibles.length>0){
                this[VisiblesID.pop()].Destroy();
                this[Visibles.pop()+"_Zoom"].Destroy();
        }
    }

        function Hola(){
                return "Hola formMostrar";
        }
}
```

6 Millores i Conclusions

6.1 Futur de l'aplicació

Amb la realització d'aquesta aplicació s'ha obert un camí en la representació dels diferents conceptes de biologia que son susceptibles de ser representats. En aquest cas s'ha optat per aquests, per motius d'acotar el treball, però hi ha molts més elements a representar. Alguns d'ells, com és el cas del CDS i exons, estan implementats a la pròpia aplicació tot i que s'ha optat per que no figuressin a la documentació per no estar totalment finalitzada la seva representació gràfica i la seva relació amb el ARN. En un futur també seria interessant ampliar el tipus de molècules a tractar, limitat actualment a l'ADN, a altres tipus com ARN.

6.2 Millores

Com a tot desenvolupament d'una aplicació, a mesura que es va realitzant es van veient possibles millores que no es poden aplicar en el moment pel seu cost (econòmic o de temps) d'implementació. Algunes de les quals comentarem a continuació:

- Revisió de l'entorn gràfic (GUI). Millorar l'aspecte, alineacions, ...
- Pantalles més intuïtives i amb més missatges d'informació.
- Millorar el tractament d'elements visibles. Optimitzar funcions i gestió de memòria per minimitzar al màxim el consum.
- Crear una taula precalculada pel posicionament dels elements. Evitant el càlcul de posició cada vegada que es vol visualitzar o fer zoom.
- Crear diferents tipus d'elements visibles (formes de representació).
- Replantejar la visualització del codi ADN, per exemple, a un TextArea i a una nova pantalla amb més capacitat.
- Visualitzar amb colors els elements visibles al codi representat en el TextArea.
- Millora del tractament de fitxer amb algoritmes més eficients (augmentant els blocs de lectura).
- Millorar la cerca d'ORF amb algoritmes que faci una sola passada per cadena. Aquesta s'hauria de fer un "entre mig" entre velocitat i consum de memòria. Un algoritme que fes una sola passada per cadena implicaria un consum de memòria important que en el cas de molècules grans podrien plantejar un problema.
- Suport a molècules de cadena simple, actualment no es diferencien.
- Afegir suport multiusuari, adaptar les taules de la Base de Dades per tenir un cap identificador d'usuari. Modificar les classes per oferir aquesta millora.
- Fer proves de rendiment dels Scripts en Perl fent una comparativa amb altres llenguatges.

- Perdent l'avantatge del llenguatge Perl amb els biòlegs, buscar el rendiment i velocitat de processament amb la traducció dels scripts a llenguatge C.
- Realització d'un entorn de descàrrega automàtica de molècules i la seva descripció de la base de dades GENBANK de NCBI.
- Realitzar un parser de XML de la informació de GENBANK per extreure automàticament elements que es poden visualitzar dins d'aquesta aplicació.
- Finalitzar l'apartat de CDS i Exons un cop afegida la compatibilitat amb molècules d'ARN.
- Millorar el suport a molècules circulars tant la representació com la cerca d'ORFs
- Assignar profunditats de visualització a cada tipus d'element per evitar que els elements més grans ocultin els petits.
- Millorar la gestió de fitxers ".fasta" amb la interpretació de les capçaleres i múltiples molècules per fitxer.
- Adaptar la motxilla de labVir per passar informació de l'eina "Visor" a les altres i a la inversa.
- Afegir una capa de seguretat en ActionScript i AMFPHP.
- Actualitzar versions de Flash, Ariware ARP, PHP i AMFPHP.
- Fer una pàgina per crear l'usuari de la Base de Dades i donar-li permisos automàticament.
- Sistema de captura d'imatges dels gràfics o generació de fitxers descarregables per poder-los comparar.
- Implementar botó dret del ratolí sobre els elements per descarregar els fragments d'ADN corresponent.

6.3 Conclusions

Cada dia és més evident la necessitat d'implantació de la informàtica en tots els àmbits, la biologia no és una excepció. Aquest treball en conjunt amb el seu predecessor labVir, han mostrat com amb l'ajut de la informàtica, es pot apropar a la gent conceptes de la biologia aparentment complicats. Oferint unes eines per entendre una mica millor, allò que es pot explicar, però difícilment mostrar si no disposes de recursos suficients.
Aquesta aplicació va néixer sabent que no havia de ser una aplicació per competir amb eines professionals, sinó que havia de ser una eina de suport a la docència, més bàsica i simplificant conceptes. Tot i això, l'entorn creat pot ser utilitzat en diversos nivells acadèmics a mode d'introducció o com a primeres aproximacions en àmbits més professionals. El fet de permetre tractar molècules inventades o no conegudes fa que pugui ser utilitzat per tenir una imatge visual de molècules en estudi.

7 Bibliografia

Desenvolupament d'un laboratori virtual per a les pràctiques de Biologia Molècula
Jordi Romero Sallent (2006)
"Memòria del Treball Final de Carrera de labVir"

Manual de PHP PHP.net
http://www.php.net/manual/es/
Consulta 2008-2009
"Manuals de PHP"

Perl Documentation
http://perldoc.perl.org/
Consulta 2008-2009
"Manuals i definicions de Perl"

Extensible Markup Language. Wikipedia
http://ca.wikipedia.org/wiki/XML
Consulta 2008-2009
"Definició del format XML"

Formato Fasta
http://es.wikipedia.org/wiki/Formato_FASTA Wikipedia
Consulta 2008-2009
"Definició del format de fitxers FASTA"

Gen
http://es.wikipedia.org/wiki/Genes Wikipedia
Consulta 2008-2009
"Definició dels gens"

ORF
http://es.wikipedia.org/wiki/Marco_abierto_de_lectura Wikipedia
Consulta 2008-2009
"Definició d' ORF"

Google
http://www.google.es/Cercador
Consulta 2008-2009
"Inici de tota cerca d'ajuda realitzada"

Flash Resources
http://www.adobe.com/support/documentation/en/flash/ Adobe
Consulta 2008-2009
“Manuals de Flash i ActionScript”

Free flex/flash and php files
http://www.sephiroth.it/ Flash
Consulta 2008-2009
“Exemple i utilitats per desenvolupament d’aplicacions en Flash i PHP”

8 Anex

8.1 Instal·lació del sistema

Cal tenir un servidor de pàgines web. En cas de no tenir-ne un instal·lat recomanem l'entorn XAMPP per integrar els serveis de pàgines web, servidor de base de dades, PHP i Perl i de fàcil instal·lació.

El podem descarregar de:
http://www.apachefriends.org/en/xampp.html

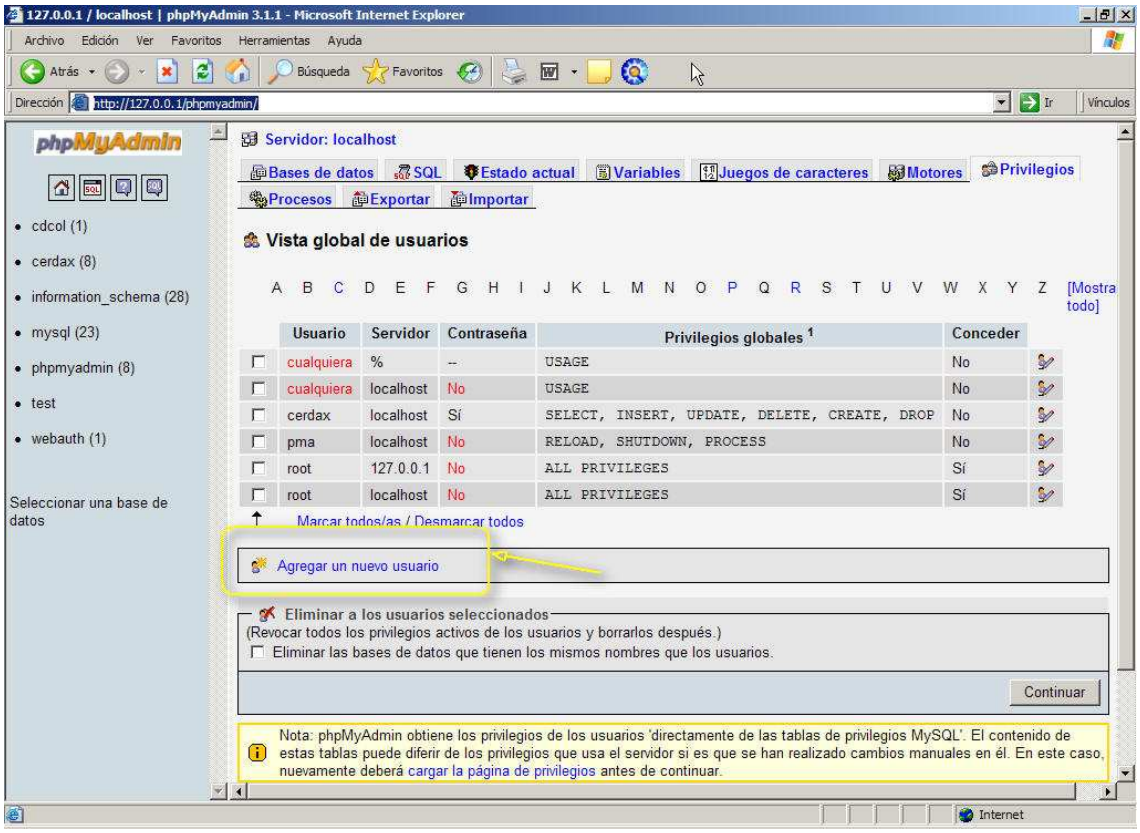
També és necessari en Windows instal·lar una versió de Perl més completa que la que porta XAMPP per això ens descarregarem ActivePerl.

http://www.activestate.com/activeperl/

Un cop instal·lat s'ha de crear un usuari a la base de dades si hem instal·lat l'entorn XAMPP disposem d'un entorn Web que ens ajudarà.

Per fer-ho obrim un navegador web i posem l'adreça :

http://127.0.0.1/phpmyadmin/



A l'apartat "Privilegios" anem a "Agregar un nuevo usuario", on crearem un usuari i li associarem una nova Base de Dades i permisos.

Descomprimim l'aplicació al directori htdocs del nostre servidor. En XAMPP i windows el trobarem a

C:\XAMPP\htdocs

Amb el nom d'usuari i password editem el fitxer de configuració amb les dades d'usuari i password creats.

C:\XAMPP\htdocs\lab\configuracio\connect.php

Per acabar la instal·lació de l'aplicació haurem d'anar a l'adreça següent que crearà l'estructura de taules.

http://localhost/lab/configuracio/instalar.php

Un cop creat ja podem accedir al sistema amb l'adreça:

http://localhost/lab/