SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 581

Web sučelje za poravnanje sljedova

Dino Blažeka

Zagreb, srpanj 2013.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujuem Miletu na iznimnom strpljenju i razumijevanju te Korparu koji je izradio biblioteku na kojoj se bazira ovaj rad

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

1. Uvod

Cilj ovog rada je izrada web sučelja za poravnanje sljedova. Web sučelje bazirano je na postojećoj biblioteci za poravnanje sljedova proteina i nukleinskih kiselina imena SW#. Ona za svoje izvršavanje koristi grafičke procesore zasnovane na CUDA arhitekturi. Web sučelje za korištenje SW# je važno iz razloga što grafičke kartice koje podržavaju CUDA arhitekturu često nisu dostupne istraživačima s područja bioinformatike pa im je korištenjem web sučelja omogućeno korištenje resursa bez potrebe za nabavkom sklopovske opreme.

Korisnicima se na raspolaganje stavljaju četiri modula koji su dio SW# biblioteke:

SW#n: služi za poravnanje jednog para dugih sljedova nukleotida koristeći Smith-Waterman algoritam, Needleman-Wunsch algoritam ili Hybrid-Waterman algoritam,

SW#nc: služi za poravnanje jednog para dugih sljedova nukleotida koristeći Smith-Waterman algoritam, Needleman-Wunsch algoritam ili Hybrid-Waterman algoritam, pretražuje i komplementarni DNA lanac,

SW#p: služi za poravnanje jednog para proteina koristeći Smith-Waterman algoritam, Needleman-Wunsch algoritam ili Hybrid-Waterman algoritam, te

SW#db: služi za poravnanje jednog proteina s bazom proteina ili slijeda nukleotida s bazom sljedova nukleotida koristeći Smith-Waterman algoritam.

U poglavlju 2 dan je kratak uvid u teoriju vezanu uz poravnanje slijedova te je iznesena osnovna terminologija. Poglavlje 3 opisuje module alata SW# te parametre koje oni primaju. Upute za korištenje dane su u poglavlju 4, a tehnologije korištene pri izradi programskog ostvarenja web sučelje i programsko ostvarenje samog web sučelja opisani su u poglavljima 5 i 6.

2. Poravnanje sljedova

Poravnanje sljedova je postupak kojim se utvrđuju sličnosti među molekulama DNK, RNK ili proteina. Te tri vrste molukula su makromolekule zajedničke svim poznatim oblicima živih organizama. Sličnosti među njima mogu biti indikator funkcionalnih, strukturnih ili evolucijskih veza među entitetima kojima sljedovi pripadaju.

Poravnanje sljedova nije postupak koji se koristi samo u bioinformatici, već se, na primjer, koristi i u procesiranju jezika te obradi financijskih podataka.

2.1. Teminologija

Uvodimo terminologiju:

abeceda Σ: skup znakova od kojih se tvore sljedovi,

a, b: sljedovi sastavljeni od znakova iz Σ,

m: duljina slijeda a,

n: duljina slijeda b.

Neka se Σ sastoji od znakova: A, T, C i G. Neka je niz a jednak ATCG, a niz b jednak AGCCGA. Dva moguća poravnanja sljedova dana su u tablici 2.1.

Tablica 2.1: Primjer dva moguća poravnanja sljedova ATCG i AGCCG

A	T	-	C	G	-
A	G	C	C	G	A
		i			
A	T	C	G	-	-
A	G	C	C	G	A

2.1.1. Ocjenjivanje kvalitete poravnanja

Postavlja se pitanje koje je od navedenih poravnanja bolje. Da bi se moglo odgovoriti potrebno je uvesti još termina. Procjep (engl. *gap*) je dio slijeda u poravnanju koji se označava uzastopnim znakovima ”-”. Broj uzastopnih znakova ”-” nazivamo duljinom procjepa. Pojavljivanje prvog znaka ”-” u procjepu nazivamo otvaranje procjepa (engl. *gap opening*), dok svaki sljedeći nazivamo produljenjem procjepa (engl. *gap extension*).Može se reći da procjep označava operacije brisanja ili umetanja. Kažemo da je došlo do umetanja ako se na nekoj poziciji u poravnatim sljedovima u slijedu a nalazi procjep, a u slijedu b znak abecede. Za operaciju brisanja vrijedi suprotno – kažemo da je došlo do operacije brisanja ako se na određenoj poziciji u nizu a nalazi znak abecede, a u nizu b procjep. Također, postoji i operacija zamjene ili supstitucije. Ona se manifestira na način da se na određenoj poziciji u poravnanju sljedova u oba slijeda nalazi znak iz abecede. Poseban slučaj supstitucije u kojem su na određenoj poziciji u oba slijeda jednaki znakovi nazivamo identitet.

Da bi ocijenili poravnanje, pretpostavimo da tablica supstitucija za iste znakove vraća ocjenu 3 (engl. *match score*), a za različite -1 (engl. *mismatch score*) te da se otvaranje i produljenje procjepa ocjenjuje s -2. Tada bi ocjena prvog poravnanja bila 3-1-2+3+3-2=4, a ocjena drugog 3-1+3-1-2-2=0. Uz dane parametre, prvo poravnanje je bolje.

2.2. Algoritmi za poravnanje sljedova

Postoje tri tipa poravnanja sljedova:

1. globalno poravnanje,
2. lokalno poravnanje, te
3. poluglobalno poravnanje.

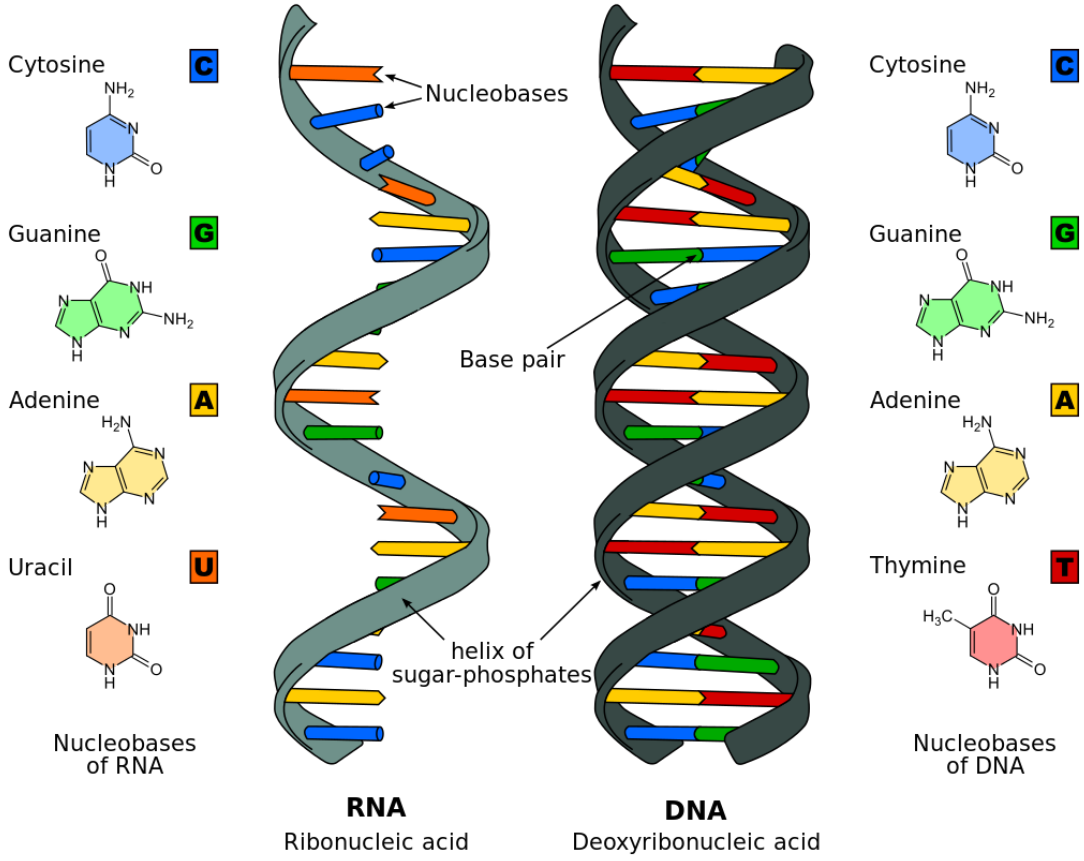
Algoritmi za globalno poravnanje pokušavaju poravnati svaki član svakog sljeda, tj. osiguravaju da poravnanje sljedova a i b sadrži sve članove slijeda a i slijeda b. Algoritmi za globalno poravnanje najčešće se koriste kod poravnanja sljedova koji su relativno slični te su otprilike jednake duljine.

Algoritmi za lokalno poravnanje češće se koriste za poravnanje sljedova među kojima postoje velike razlike, ali se za njih pretpostavlja da sadrže regije čija je sličnost velika. Kod njih poravnanje ne mora sadržavati sve članove sljedova a i b.

Algoritmi za poluglobalno poravnanje pokušavaju naći najbolje moguće poravnanje koje sadrži sve članove slijeda a, a ne mora sadržavati sve članove slijeda b. Nastaju kao mješavina algoritama za lokalno i algoritama za globalno poravnanje te tako čine kompromis između njih. Jedan primjer njihovog korištenja je onaj u kojem je slijed a puno kraći od slijeda b te poravnanjem želimo pronaći regiju slijeda b najsličniju slijedu a.

2.3. Deoksiribonukleinska kiselina – DNK

DNK ili deoksiribonukleinska kiselina (engl. *DNA, deoxyribonucleic acid*) je molekula kojom su kodirane genetske upute živih organizama. DNK je oblika dvostruke spiralne zavojnice kako je prikazano na slici 2.1.



Slika 2.1: Usporedba molekule DNK i RNK [1]

Svaki od dva lanca je niz nukleotida. Oni su građeni od:

- nukleinske baze,
- deoksiriboze, te

– jedne ili više fosfatnih grupa.

Nukleinske baze, prema kojima razlikujemo nukleotide i koje se javljaju u DNK su:

A: adenin,

T: timin,

C: citozin, te

G: gvanin.

One ujedno čine abecedu kod poravnanja molekula DNK. Dva lanca DNK međusobno su povezani vodikovim vezama koje su raspoređene tako da vežu komplementarne baze. Adenin je komplementaran timinu i obrnuto, dok je citozin komplement gvaninu. Primjerice, komplement lanca ATCG je lanac TAGC.

2.4. Ribonukleinska kiselina – RNK

RNK ili ribonukleinska kiselina (engl. *DNA, deoxyribonucleic acid*) je molekula koja obavlja važne zadaće u živim organizmima kao što su kodiranje i dekodiranje DNK. Za razliku od DNK, RNK je jednostruka molekula, tj. sastoji se od samo jednog lanca kao što je vidljivo na slici 2.1. Lanac je, kao i kod DNK, građen od nukleotida uz razliku da su nukleotidi RNK građeni od šećera riboze umjesto deoksiriboze kao što je to slučaj s DNK. Nukleinske baze, prema kojima razlikujemo nukleotide i koje se javljaju u RNK su[3]:

A: adenin,

U: uracil,

C: citozin, te

G: gvanin.

One ujedno čine abecedu kod poravnanja molekula RNK. Kako se umjesto timina kod RNK javlja uracil, komplementarne baze su adenozin i uracil te citozin i gvaninu. Primjerice, komplement lanca AUCG je lanac UAGC.

2.5. Proteini

Bjelančevine ili proteini su makromolekule koje se sastoje od jednog ili više lanaca aminokiselina. Postoji dvadesetak aminokiselina koje tvore proteine u živim organizmima. Primarna struktura proteina zapisuje se kao niz aminokiselina koje ih tvore. Ona se koristi kao slijed, dok su abeceda znakovi koji predstavljaju aminokiseline.

3. SW#

Biblioteka SW# pruža jednostavan način za brzo izvođenje algoritama za poravnanje slijedova. Iako je sama biblioteka i nazvana po algoritmu Smith-Waterman koji je algoritam za lokalno poravnanje, u nekim modulima korisnik pomoću parametra može odabrati druge algoritme za vršenje poravnanja – Needleman-Wunsch algoritam za globalno poravnanje ili Hybrid-Waterman algoritam za poluglobalno poravnanje. Korisnicima web sučelja na raspolaganju su četiri modula koji koriste funkcionalnost biblioteke:

3.1. Modul SW#n

Modul SW#n služi za poravnanje jednog para dugih sljedova nukleotida. Izvođenje modula može se prilagoditi potrebama korisnika odabirom parametara navedenih u tablici 3.3.

Tablica 3.1: Parametri modula SW#n

Ime	Kratica	Opis
Fasta Query File	i	Putanja do datoteke sa slijedom a
Fasta Target File	j	Putanja do datoteke sa slijedom b
Gap opening penalty	g	Ocjena za otvaranje procjepa
Gap extension penalty	e	Ocjena za produljenje procjepa
Match	match	Ocjena za supstituciju jednakim znakom
Mismatch	mismatch	Ocjena za supstituciju različitim znakom
Out	out	Lokacija datoteke u koju se zapisuju rezultati
Output Format	outfmt	Format zapisa rezultata
Algorithm	alg	Odabir algoritma za poravnanje: SW, NW ili HW

3.2. Modul SW#nc

Modul SW#nc služi za poravnanje jednog para dugih sljedova nukleotida. Prima iste parametre kao i modul SW#n, ali kod poravnanja uzima u obzir i komplementarne lance nukleinske kiseline.

3.3. Modul SW#p

Modul SW#p služi za poravnanje jednog para proteina. U tu svrhu koristi se Smith-Watermanov algoritam za lokalno poravnanje. Izvođenje modula može se prilagoditi potrebama korisnika odabirom parametara navedenih u tablici 3.2.

Tablica 3.2: Parametri modula SW#p

Ime	Kratica	Opis
Fasta Query File	i	Putanja do datoteke sa slijedom a
Fasta Target File	j	Putanja do datoteke sa slijedom b
Gap opening penalty	g	Ocjena za otvaranje procjepa
Gap extension penalty	e	Ocjena za produljenje procjepa
Substitution matrix	m	Matrica s ocjenama supstitucije
Out	out	Lokacija datoteke u kou se zapisuju rezultati
Output Format	outfmt	Format zapisa rezultata
Algorithm	alg	Odabir algoritma za poravnanje: SW, NW ili HW

3.3.1. Matrica s ocjenama supstitucije

Kao ocjena supstitucije ne mora se uvijek koristiti fiksirana, već se može koristiti vrijednost iz matrice s ocjenama supstitucije M. Ocjena supstitucije tada je zadana kao element matrice na poziciji $M(x, y)$ gdje su x i y znakovi u sljedovima a i b na poziciji za koju se želi odrediti ocjena supstitucije. Matrice za ocjenu supstitucije su obično simetrične s obzirom na glavnu dijagonalu što za posljedicu ima to da je, primjerice, ocjena ocjena kada dođe do zamjene aminokiseline alanina s valinom jednaka ocjeni kod zamjene valina alaninom. Za poravnanje proteina, obično se koriste supstitucijske matrice BLOSUM koje su objavljene u radu [4], a prikazane su slikom 3.1.

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Slika 3.1: Matrica supstitucije BLOSUM62 [1]

3.4. Modul SW#db

Modul SW#db služi za poravnanje jednog proteina s bazom proteina koristeći Smith-Watermanov algoritam ili slijeda nukleotida s bazom slijedova nukleotida. U tu svrhu koristi se Smith-Watermanov algoritam za lokalno poravnanje. Za poravnanje lanaca nukleotida potrebno je kao matricu supstitucije odabrati $EDNA_FULL$ Izvođenje modula može se prilagoditi potrebama korisnika odabirom parametara navedenih u tablici 3.3.

Tablica 3.3: Parametri modula SW#db

Ime	Kratica	Opis
Fasta Query File	i	Putanja do datoteke sa slijedom aminokiselina
Fasta Target File	j	Putanja do datoteke s bazom proteina
Gap opening penalty	g	Ocjena za otvaranje procjepa
Gap extension penalty	e	Ocjena za produljenje procjepa
Substitution matrix	m	Matrica s ocjenama supstitucije
Evalue threshold	e	Prag E vrijednosti
Out	out	Lokacija datoteke u kou se zapisuju rezultati
Output Format	outfmt	Format zapisa rezultata
Algorithm	alg	Odabir algoritma za poravnanje

3.4.1. E vrijednost

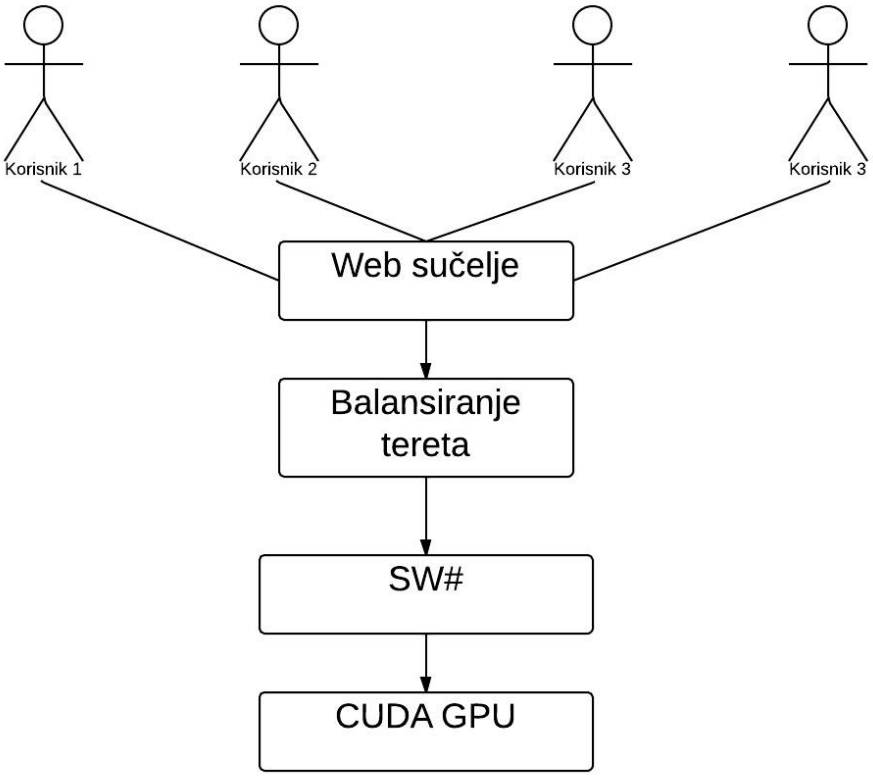
Neka je s ocjena poravnanja. Tada je E vrijednost broj nezavisnih poravnanja, s vecom ili jednakom ocjenom od s, za koja se ocekuje da će se dogoditi pri poravnanju s odredenom stopom slučajnosti. Pojednostavljeno, manja E vrijednost označava značajnije poravnanje. E vrijednost računa se izrazom 3.1.

$$E(s) = mnKe^{-\lambda s} \tag{3.1}$$

Parametri K i λ su unaprijed izračunati statistički parametri.[5]

4. Web sučelje i njegovo korištenje

Web sučeljem je udaljenim korisnicima omogućeno korištenje modula biblioteke SW# te samim time i korištenje grafičkih procesora zasnovanih na CUDA arhitekturi kako je prikazano slikom 4.1

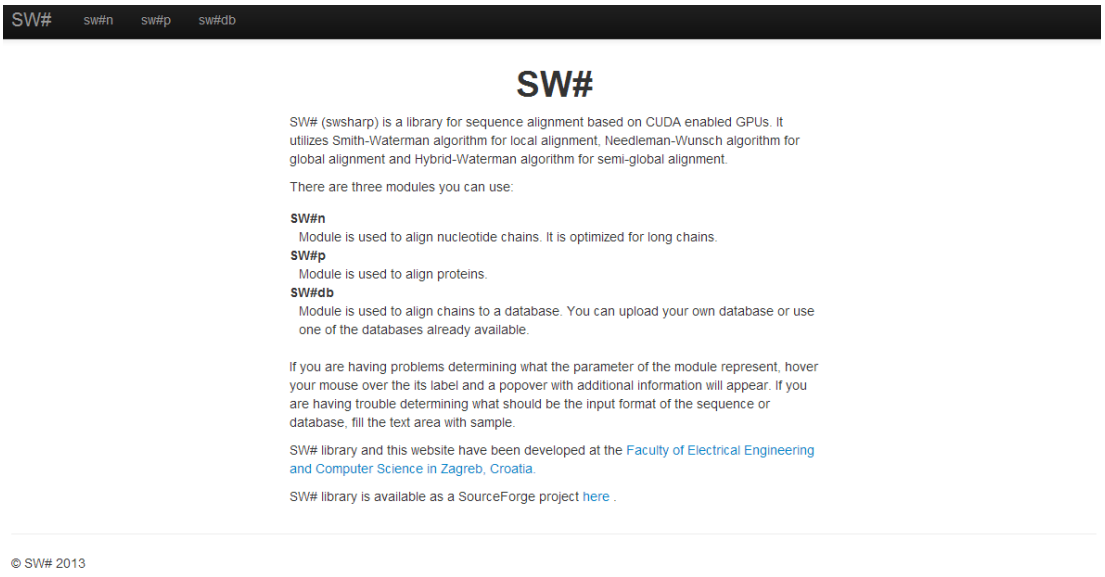


Slika 4.1: Web sučelje omogućava korisnicima izvođenje poravnanja na udaljenom poslužitelju s CUDA grafičkim karticama

Unošenjem URL-a na kojem se nalazi izrađeno sučelje, korisnik se nalazi na početnoj stranici prikazanoj na slici 4.2. Na vrhu stranice nalazi se traka kojom se odabire željeni modul.

U glavnom dijelu stranice svakog od modula nalazi se forma. Pomoću te forme

korisnik unosi parametre poravnanj koji su navedeni u tablicama u poglavlju 3. Treba napomenuti da u web sučelju ne postoji zasebna stranica za modul SW#nc već je za njegovo korištenje potrebno uključiti opciju "Calculate for complement" u web sučelje za modul SW#n. Također, umjesto unošenja baze proteina kao parametra j za modul SW#db, moguće je odabrati jednu od baza koje se već nalaze pohranjene na čvrstom disku poslužitelja.

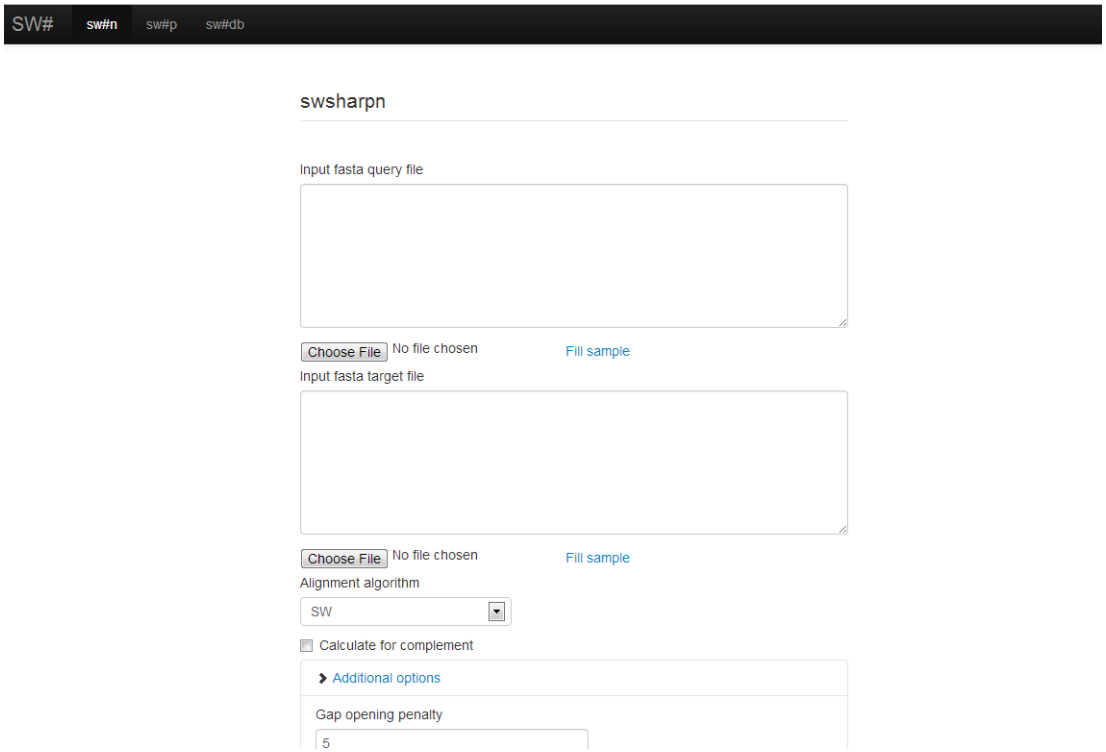


Slika 4.2: Početna stranica web sučelja

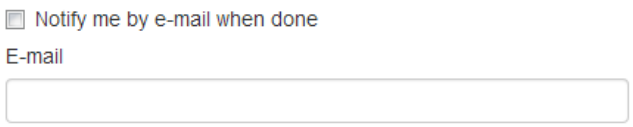
Na dnu forme svakog od modula nalazi se dio forme u kojem korisnik, ako želi, može uključiti slanje obavijesti o završetku na e-mail adresu (slika 4.4). Ako je opcija uključena, korisniku po završetku zadanog posla na prethodno zadanu e-mail adresu stiže potvrda o završetku zajedno s poveznicom na prikaz rezultata poravnanja.

Po završetku poravnanja korisniku se prikazuje web stranica koja sadrži rezultate. Stranica s rezultatima podijeljena je u dva dijela. U prvom se prikazuju rezultati poravnanja u tekstualnom obliku – koristeći pairstat format (za module SW#n i SW#p, na slici 4.5) ili tablični prikaz (za modul SW#db, na slici 4.6). U gornjem desnom kutu nalazi se poveznica za preuzimanje rezultata u formatu pairstat za module SW#n i SW#p te u formatu BLAST m 9 za modul SW#db.

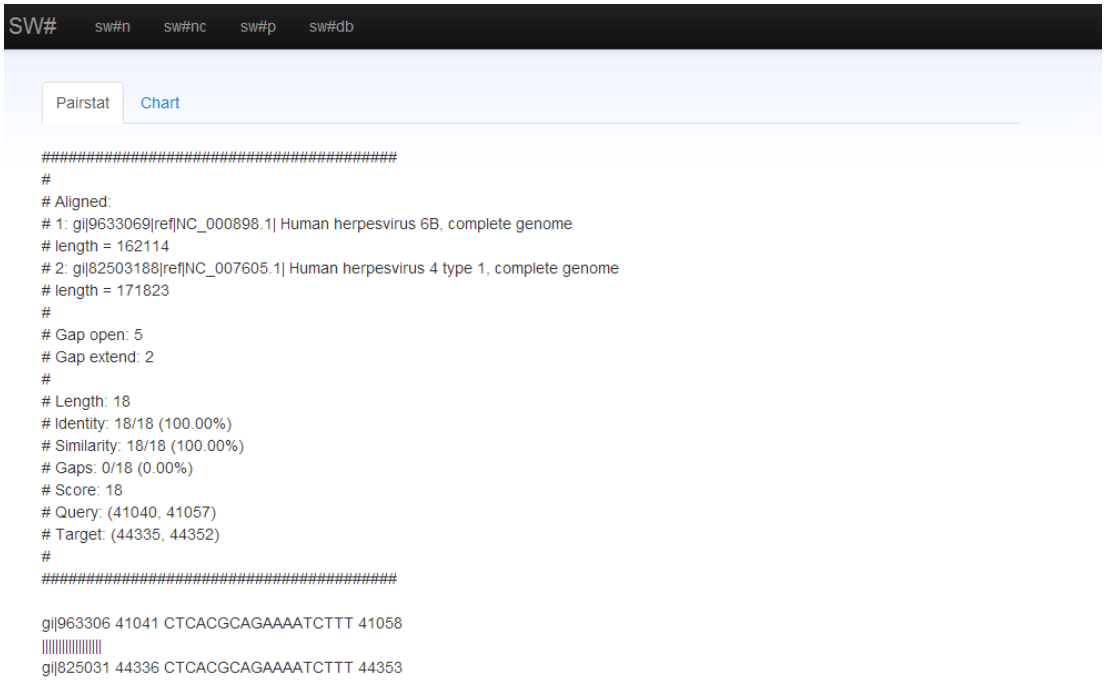
U drugom dijelu nalazi se grafički prikaz rezultata (slika 4.7). Kod rezultata poravnanja modulom SW#db ne postoji grafički prikaz rezultata iz razlog što nema praktičnog načina prikazivanja poravnanja. Korisnik može po volji umanjivati i uvećavati grafički prikaz te ga pomicati u bilo kojem smjeru x ili y osi.



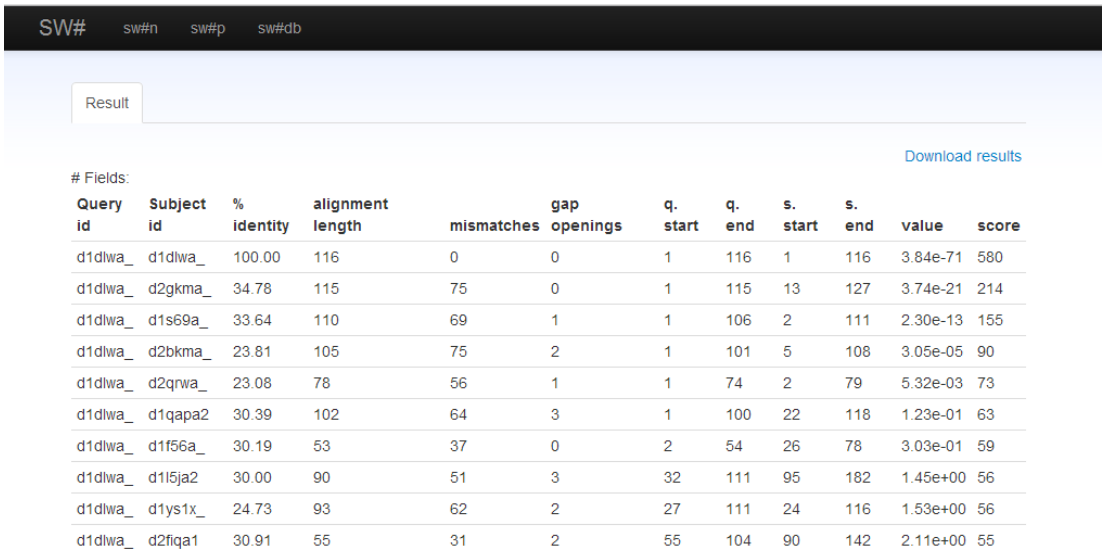
Slika 4.3: Stranica za modul SW#n



Slika 4.4: Polja za uključivanje i isključivanje obavijesti o završetku putem e-maila

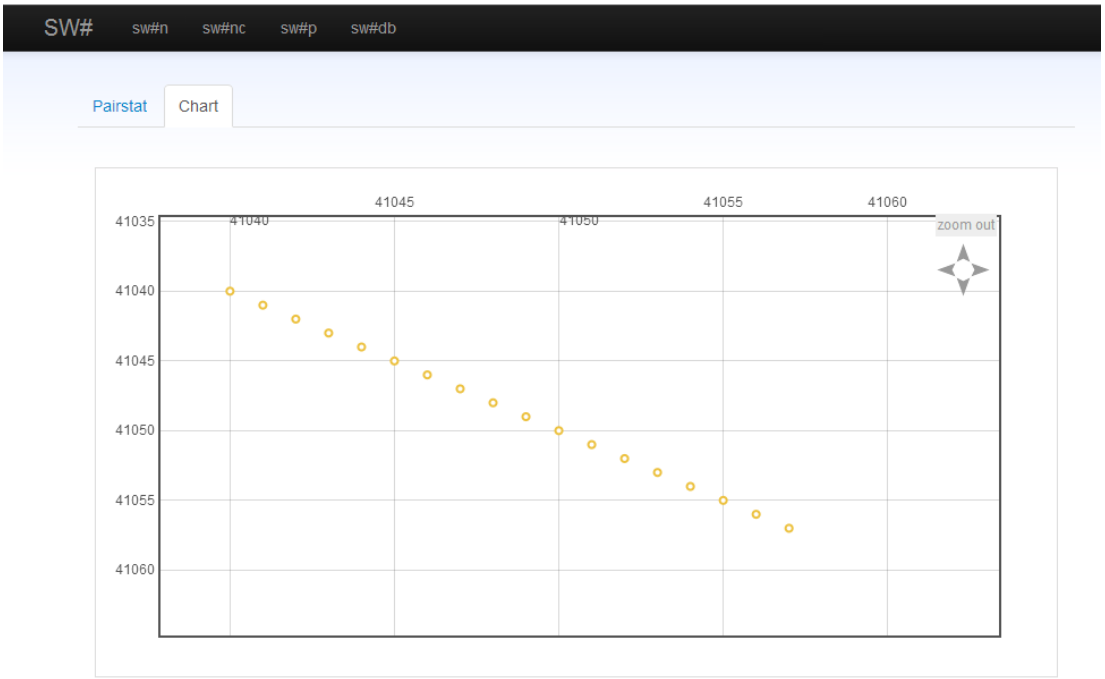


Slika 4.5: Prikaz rezultata - pairstat



Query id	Subject id	% identity	alignment length	mismatches	gap openings	q. start	q. end	s. start	s. end	value	score
d1dlwa_	d1dlwa_	100.00	116	0	0	1	116	1	116	3.84e-71	580
d1dlwa_	d2gkma_	34.78	115	75	0	1	115	13	127	3.74e-21	214
d1dlwa_	d1s69a_	33.64	110	69	1	1	106	2	111	2.30e-13	155
d1dlwa_	d2bkma_	23.81	105	75	2	1	101	5	108	3.05e-05	90
d1dlwa_	d2qrwa_	23.08	78	56	1	1	74	2	79	5.32e-03	73
d1dlwa_	d1qapa2	30.39	102	64	3	1	100	22	118	1.23e-01	63
d1dlwa_	d1f56a_	30.19	53	37	0	2	54	26	78	3.03e-01	59
d1dlwa_	d1l5ja2	30.00	90	51	3	32	111	95	182	1.45e+00	56
d1dlwa_	d1ys1x_	24.73	93	62	2	27	111	24	116	1.53e+00	56
d1dlwa_	d2fiqa1	30.91	55	31	2	55	104	90	142	2.11e+00	55

Slika 4.6: Prikaz rezultata - tablični prikaz rezultata SW#db



Slika 4.7: Prikaz rezultata - graf

5. Korištene tehnologije

Pri izradi ovog rada korišteno je mnogo tehnologija iz područja Jave i JavaScripta. U nastavku su opisane najviše korištene biblioteke i tehnologije.

5.1. Spring

Spring je biblioteka za programski jezik Javu osmišljen od strane Roda Johnsona 1992. godine kada je predstavljen uz knjigu istog autora naslova *Expert One-on-One J2EE Design and Development*. Biblioteka stječe veliku popularnost u programerskim krugovima iz razloga što pruža alternativu EJB (Enterprise JavaBean) modelu. On je nastao 1997. u okviru IBM-a te je 1999. usvojen od strane Sun Microsystemsa koji je u to vrijeme bio zadužen za razvoj Java platforme, a kasnije je kupljen od strane Oraclea. Neki od razloga zbog kojih je Spring preuzeo primat su:

- iznimne mogućnosti u korištenju oblikovnog obrasca inverzije kontrole,
- princip da je *konvencija važnija od konfiguracije*,
- lak pristup bazama podataka,
- konfiguracija pomoću anotacija kojom se izbjegava pretjerana verboznost,
- olakšano testiranje pomoću unit i integracijskih testova itd.

Danas Spring razvija kompanija Springsource koja je u vlasništvu VMWare-a. Razvijaju se mnogi projekti kao što su:

- Spring Core,
- Spring Mobile,
- Spring Web Flow,
- Spring Data,
- Spring Security, itd.

U izradi ovog rada najviše su korišteni Spring Core modul i Spring MVC modul.

5.1.1. Spring Core

Kao što mu i samo ime govori, Spring Core je jezgra svih ostalih Spring projekata. Najvažnija zadaća mu je omogućiti jednostavno i modularno korištenje oblikovnog obrasca inverzija kontrole (engl. *inversion of control, dependency injection (Ioc, DI)*). Njegov ključni dio je Spring IoC kontejner koji preuzima zadaću stvaranja objekata te umetanje drugih objekata, o kojima stvoreni objekti ovise, u njih.

Oblikovni obrazac inverzija kontrole

Inverzija kontrole je oblikovni obrazac u kojem objekti definiraju svoju ovisnost o drugim projektima jedino kroz:

1. argumente konstruktora,
2. argumente metodi tvornici (engl. *factory method*) ili
3. svojstva koja se postavljaju na objektu nakon njegovog instanciranja.

Pomoću te definicije kontejner u kojem se drže objekti ubacuje (engl. *injects*) objekte o kojima stvoreni objekt ovisi. Odatle i dolazi naziv oblikovnog obrasca *dependency injection*.

Spring IoC kontejner

Spring IoC kontejner u sebi sadrži objekte te se brine o njihovim ovisnostima. Kontejneru se zadaju upute kako da stvara, povezuje i konfigurira objekte. Upute su metapodaci koji mogu biti zadani pomoću:

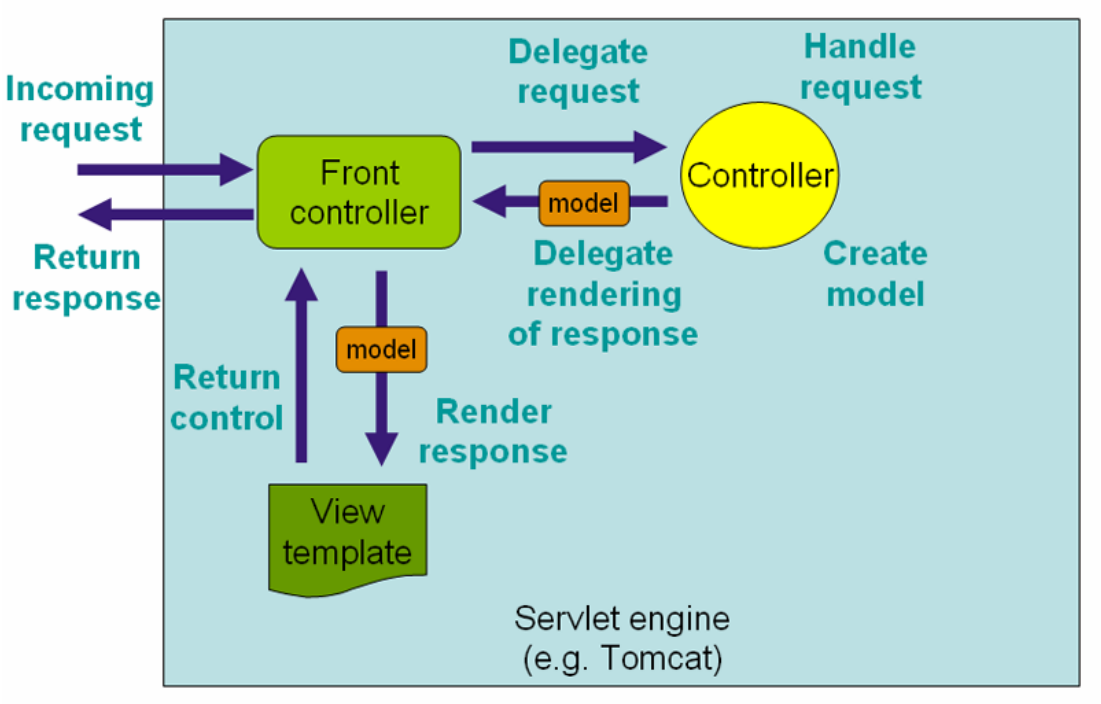
1. XML konfiguracijske datoteke,
2. Java anotacija ili
3. u samom Java kodu.

Ovakav način konfiguriranja nudi korisniku mogućnost da definira objekte koji čine aplikacije te njihovu kompleksnu međuspregu.

Spring Web MVC

Spring Web MVC je modul Springa čiji je središnji dio *DispatcherServlet*. On raspodijeljuje zahtjeve kontrolerima na obradu te dinamički razrješava poglede kako je

prikazano na slici 5.1. Oblikovni obrazac koji se koristi je *Front controller*, a *DispatcherServlet* preuzima ulogu *front controllera*. Nakon primitka HTTP zahtjeva, *DispatcherServlet* prosljeđuje zahtjev odgovarajucem kontroleru. Izbor odgovarajućeg kontrolera vrši se na temelju URL-a na koji je zahtjev poslan te parametrima koji su poslani u samom zahtjevu. Kontroler tada vraća model. *DispatcherServlet* iz konfiguracije očitava odgovarajući *ViewResolver* te mu prosljeđuje model. *ViewResolver* na temelju modela iscrtava pogled (engl. *view*) koji se preko *DispatcherServlet*-a vraća pošiljatelju zahtjeva.



Slika 5.1: Prikaz rada Spring Web MVC-a[2]

5.2. JavaServer Pages Standard Tag Library

JavaServer Pages Standard Tag Library (JSTL) je biblioteka tagova koji se mogu koristiti unutar JavaServer Pagesa (JSP). Verzija 1.2 izdana je 2006. godine. Tagovi omogućuju sažeto pisanje koda za kompleksne i često korištene mogućnosti unutar JSP-a. Na taj način izbjegava se korištenje Java koda unutar JSP-a. Neke od mogućnosti koje nudi JSTL biblioteka su:

- procesiranje XML-a,
- iteracija po raznim kolekcijama podataka,

– lak prijenos podataka do pogleda (engl. *View*),

– internacionalizacija,

– mogućnost implementacije vlastitih tagova, itd.

5.3. Maven

Maven je proizvod otvorenog koda kojeg razvija Apache. Maven služi za automatizaciju procesa izgradnje aplikacije (engl. *build automation tool*). Njegovo podešavanje obavlja se kroz datoteke u XML formatu. Maven također brine o ovisnosti projekta o ostalim projektima te njihovim artefaktima (npr. jar ili war). Svaki artefakt koji se nalazi u centralnom Maven repozitoriju dostupan je svim korisnicima te je podržano njegovo verzioniranje. Na taj način korisnik je oslobođen mukotrpnog posla vođenja brige o ovisnostima svog projekta u kojem se iznimno lako rade greške koje se ponekad manifestiraju tek kada je kod pokrenut, a ne u ranijim fazama prevođenja ili izgradnje, što može izazvati velike probleme.

5.4. JavaScript

JavaScript je interpretirani programski jezik. Brendan Eich ga je osmislio 1996. godine u istraživačkom odjelu tvrtke Netscape, koja je u to vrijeme bila poznata po svom web pregledniku Netscape Navigator. Prvotna namjena bila mu je izvršavanje skripti na klijentskoj strani da bi se poboljšala interakcija tadašnjih web stranica i korisnika. I danas se najčešće koristi za tu namjenu ali sve više raste broj biblioteka koje ga čine pogodnim za izvršavanje na poslužiteljskoj strani. Jedan od primjera takve biblioteke je Node.js. Unatoč svom imenu, JavaScript nema mnogo dodirnih točaka s Javom. Prvotno ime bilo mu je LiveScript, ali kako se njegovo izdavanje poklopilo s ugrađivanjem podrške za Javu u Netscape Navigator, ime je promijenjeno u JavaScript.

Neke od značajki jezika su da je:

Imperativan i strukturiran: sintaksa petlji i uvjeta vrlo je slična onoj programskog jezika C, uz jednu veliku razliku – doseg varijabli nije ograničen viticama, već funkcijama,

Dinamički pisan (engl. *dynamic typing*): tipovi su vezani uz vrijednosti ,a ne uz varijable,

Funkcijski: funkcije su i same objekti te je nad njima moguće pozivati druge funkcije,

Prototipni: za razliku od većine ostalih objektno orijentiranih jezika (npr. Jave) koji su objektni klasni jezici, JavaScript je objektni prototipni jezik.

5.4.1. jQuery

JQuery je jedna od mnogih biblioteka pisanih u JavaScriptu koje omogućavaju izvršavanje kompleksnih akcija u JavaScriptu. Neke od njih su:

- prolaz HTML dokumentom i njegovo pretraživanje,
- dinamička izmjena HTML dokumenta,
- slanje Ajax zahtjeva prema poslužitelju te primanje odgovora,
- animacija,
- podrška za razne web preglednike, itd.

6. Programsko ostvarenje

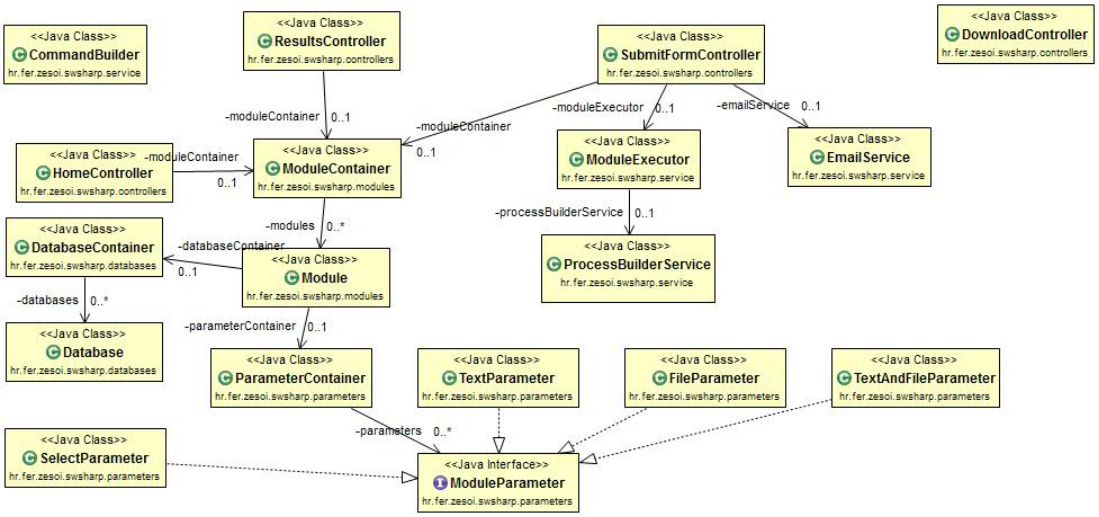
Poslužiteljska strana web sučelja ostvarena je u programskom jeziku Java. Kod je raspodijeljen u 5 paketa:

1. hr.fer.zesoi.swsharp.controller,
2. hr.fer.zesoi.swsharp.chart,
3. hr.fer.zesoi.swsharp.databases,
4. hr.fer.zesoi.swsharp.modules,
5. hr.fer.zesoi.swsharp.parameters,
6. hr.fer.zesoi.swsharp.service.

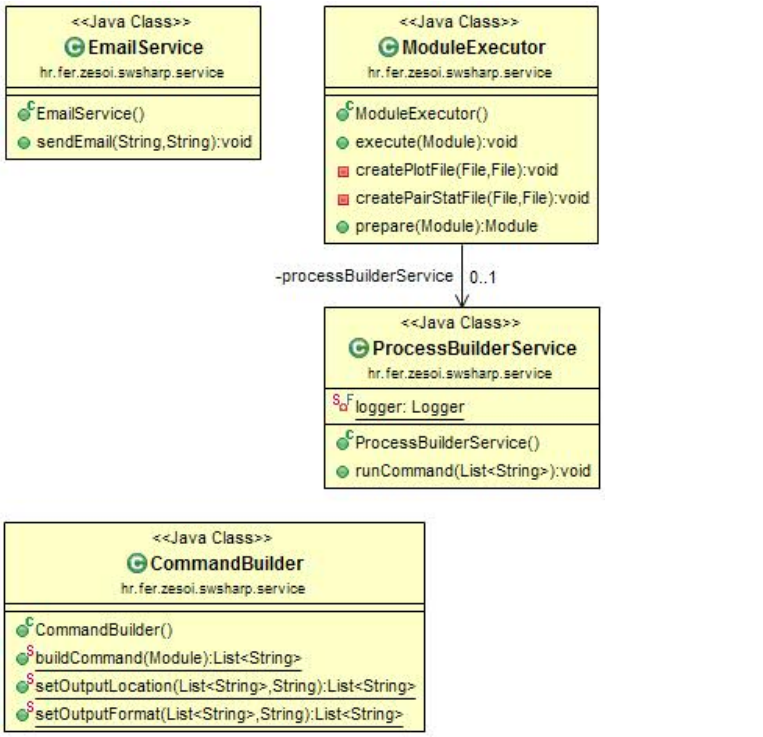
Pogledi (engl. *Views*), pisani su kao JavaServer Pages u kojima se koristi HTML, JavaScript i JSTL, izdvojeni su u zasebni direktorij *views* koji se nakon postavljanja aplikacije na poslužitelj automatski postavlja u putanju poslužitelja te mu time postaje dostupan.

6.1. Paket hr.fer.zesoi.swsharp.service

U paketu *hr.fer.zesoi.swsharp.service* naalaze razredi koji omogućavaju izvršavanje pozadinskih radnji kao što su pokretanje procesa operacijskog sustava ili slanje e-maila.



Slika 6.1: Class dijagram s važnijim razredima



Slika 6.2: Class dijagram paketa hr.fer.zesoi.swsharp.service

6.1.1. Razred ProcessBuilderService

Razred *ProcessBuilderService* omogućava pokretanje procesa operacijskog sustava. Ima jednu metodu potpisa

```
public synchronized void runCommand(List<String> command)
```

Metoda runCommand kao parametar prima listu znakovnih nizova. Ako lista sadrži tri člana:

1. pwd,
2. >,
3. out,

pokretanjem metode pokreće se proces operacijskog sustava ekvivalentan unosu naredbe *pwd* > *out* u konzolu samog operacijskog sustava. Pokretanjem gornjeg primjera u datoteku out ispisuje se naziv trenutnog radnog direktorija.

Balansiranje tereta

Mjerenja brzine pokazala su da se poravnanja pomoću biblioteke SW# izvode brže i efikasnije ako se zadaci poravnanja pokreću jedan za drugim umjesto više njih istovremeno. Modifikator *synchronized* u potpisu metode *runCommand* omogućava to da samo jedna dretva uđe u odsječak koda koji pokreće module SW#-a pa je time izbjegnuto neefikasno izvođenje poravnanja.

6.1.2. Razred ModuleExecutor

Namjena razreda *ModuleExecutor* jest iščitavanje parametara pokretanja modula te pokretanje samog modula. Metoda *execute* potpisa

```
public void execute(Module module)
```

iz modula iščitava vrijednosti parametara koje je zadao korisnik te uz pomoć razreda *ProcessBuilderService* pokreće odgovarajući proces operacijskog sustava. Ovaj razred vrši i niz predradnji metodom

```
public Module prepare(Module module)
```

koje je nužno izvesti prije samog pokretanja modula.

Pokretanjem modula rezultati se ispisuju u *dump* datoteku da bi se nakon toga pomoću modula SW#out iz nje mogli dobiti korisni podaci – pairstat ili Blast m 9 ispis te podaci o grafu kojeg je potrebno prikazati – bez ponovnog pokretanja zadatka poravnanja čime se štedi vrijeme.

6.1.3. Razred CommandBuilder

Razred *CommandBuilder* je pomoćni razred kojeg koristi *ModuleExecutor* da bi izgradio potrebnu komandu za pokretanje modula kao procesa operacijskog sustava. Razred također pruža pomoćne metode

```
public static List<String> setOutputLocation
public static List<String> setOutputFormat
```

koje omogućavaju odabir lokacije izlazne datoteke te njezin format za već postojeću naredbu pokretanja modula.

6.1.4. Razred EmailService

Razred *EmailService* nudi jednu metodu potpisa

```
public void sendEmail(String address, String resultsUrl).
```

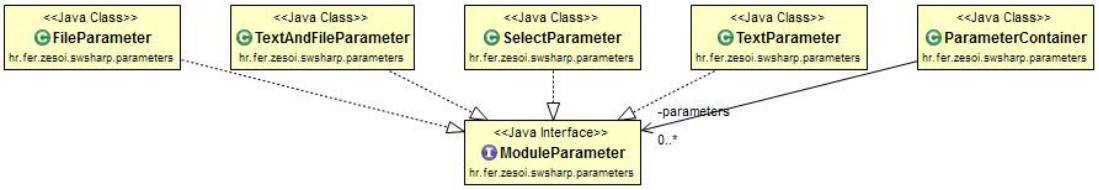
Ona šalje e-mail na e-mail adresu *address* o tome da je završeno poravnanje. U e-mailu je uključena poveznica na rezultate poravnanja *resultsUrl*.

6.2. Paket hr.fer.zesoi.swsharp.parameters

Paket *hr.fer.zesoi.swsharp.parameters* sadrži pet razreda i jedno sučelje. Oni služe za opisivanje parametara kojima se pokreću moduli SW# biblioteke.

6.2.1. Sučelje ModuleParameter

Sučelje *ModuleParameter* propisuje metode koje svaki parametar modula mora imati. Sve navedene metode moraju biti javne, a mogu se naći u tablici 6.1.



Slika 6.3: Class paketa hr.fer.zesoi.swsharp.parameters

6.2.2. Razred ParameterContainer

Razred *ParameterContainer* služi kao kontejner u kojem se drže parametri određenog modula.

Tablica 6.1: Metode sučelja *ModuleParameter*

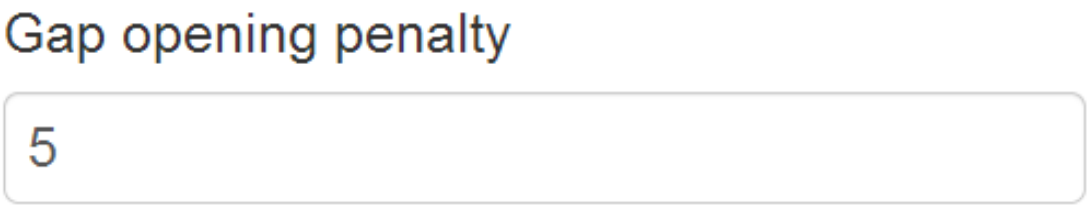
Povratni tip	Ime	Opis
String	*getName()*	Ime parametra u formi
String	*getType()*	Tip parametra
String	*getDisplayName()*	Prikaz labele u formi
String	*getCommandArgumentName()*	Ime argumenta unutar naredbe
String	*getCommandArgument()*	Vrijednost argumenta unutar naredbe
void	*setValue(Object value)*	Postavljanje vrijednosti parametra
String	*getDefaultValue()*	Dohvat izvorne vrijednosti parametra
void	*setDefaultValue(String value)*	Postavljanje izvorne vrijednosti parametra
boolean	*getHidden()*	Je li parametar skriven
void	*setHidden(boolean hidden)*	Postavljanje skriven/prikazan

6.2.3. Razred TextParameter

Razred *TextParameter* služi kao opisnik tekstualnih parametara modula. U Spring kontekstu može se dodati kao njegova instanca se dodaje i konfigurira na sljedeći način:

```
<bean id="swsharpng" class="hr.fer.zesoi.swsharp.
    parameters.TextParameter"
  scope="prototype">
  <property name="displayName" value="Gap opening penalty
     " />
  <property name="name" value="swsharpng" />
  <property name="commandArgumentName" value="-g" />
  <property name="hidden" value="true" />
  <property name="defaultValue" value="5" />
  <property name="tooltipText"
    value="Gap opening penalty, must be given as a
       positive integer" />
</bean>
```

Takva konfiguracija će se u formi za zadavanje parametara modula manifestirati kako je prikazano na slici 6.4.



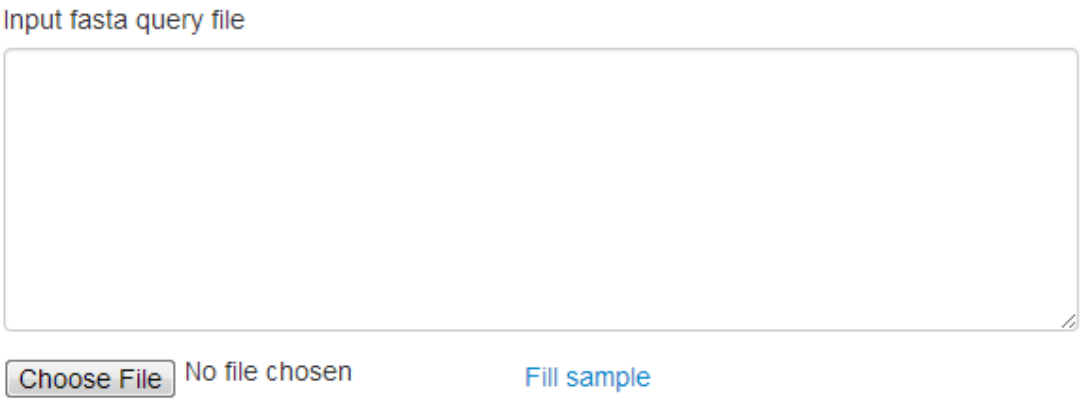
Slika 6.4: Prikaz *TextParameter*-a u web sučelju

6.2.4. Razred TextAndFileParameter

Razred *TextAndFileParameter* služi kao opisnik parametara kojima se u module unose sljedovi. On bi mogao biti i tekstualni parametar ali se ovim rješenjem pomaže korisniku da lakše unese željene sljedove. Tako korisnik sljedove može unositi popunjavanjem tekstualnog polja ili predati slijed kao datoteku. U Spring kontekst instanca razreda *TextAndFileParameter* može se dodati i konfigurirati:

```
<bean id="swsharpni" class="hr.fer.zesoi.swsharp.
    parameters.TextAndFileParameter"
        scope="prototype">
  <property name="displayName" value="Input fasta query
      file" />
  <property name="name" value="swsharpni" />
  <property name="commandArgumentName" value="-i" />
  <property name="tooltipText"
    value="Query file containing list of nucleotides in
       FASTA format" />
</bean>
```

Takva konfiguracija će se u formi za zadavanje parametara modula manifestirati kako je prikazano na slici 6.5.



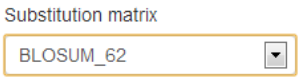
Slika 6.5: Prikaz *TextAndFileParameter*-a u web sučelju

6.2.5. Razred SelectParameter

Razred *SelectParameter* služi kao opisnik parametara koji se u web sučelju prikazuju kao padajući izbornici. Instanca *SelectParameter* razreda može se dodati u Spring kontekst i konfigurirati na sljedeći način:

```
<bean id="swsharppm" class="hr.fer.zesoi.swsharp.
    parameters.SelectParameter"
  scope="prototype">
  <property name="displayName" value="Substitution matrix
     " />
  <property name="name" value="swsharppm" />
  <property name="commandArgumentName" value="-m" />
  <property name="options"
    value="BLOSUM_45, BLOSUM_50, BLOSUM_62, BLOSUM_80,
       BLOSUM_90, PAM_30, PAM_70, PAM_250" />
  <property name="defaultValue" value="BLOSUM_62" />
  <property name="hidden" value="true" />
  <property name="tooltipText" value="Substitution matrix
      used by algorithm" />
</bean>
```

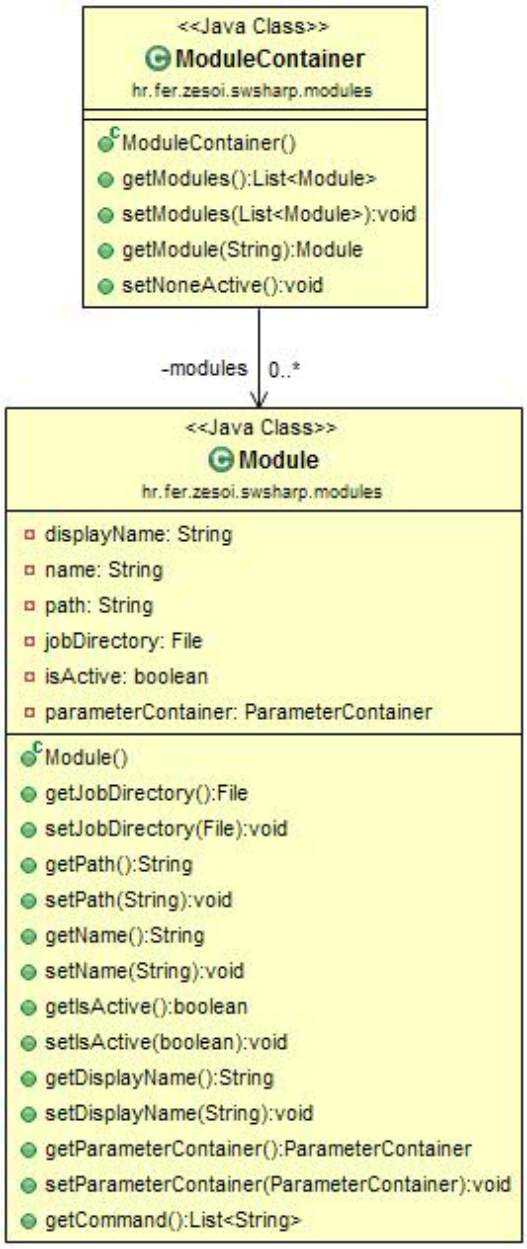
Takva konfiguracija će se u formi za zadavanje parametara modula manifestirati kako je prikazano na slici 6.6.



Slika 6.6: Prikaz *SelectParameter*-a u web sučelju

6.3. Paket hr.fer.zesoi.swsharp.modules

Paket *hr.fer.zesoi.swsharp.modules* sadrži dva razreda. Razred *ModuleContainer* služi kao kontejner u kojem se drže svi podržani moduli.



Slika 6.7: Class dijagram paketa hr.fer.zesoi.swsharp.modules

Razred *Module* služi kao opisnik modula iz SW# biblioteke. Opisi polja koje sadrži modul dana su u tablici 6.2

Tablica 6.2: Polja razreda *Module*

Tip	Polje	Opis
String	name	Ime modula
String	displayName	Ime modula koje se prikazuje na web-stranici
String	path	Putanja za pokretanje modula
File	jobDirectory	Direktorij u koji se spremaju parametri
boolean	isActive	Određuje da prikazuje li se na web stranici
ParameterContainer	parameterContainer	Sadrži opise parametara modula

Moduli te njihov kontejner zadaju se pomoću XML konfiguracijske datoteke Spring Ioc kontejnera (tzv. Spring kontekst). Uzmimo za primjer modul SW#n. Konfiguracija pripadajućeg instance razreda *Module* unutar Spring konteksta izgleda kako slijedi

```
<bean id="swsharpn"
class="hr.fer.zesoi.swsharp.modules.Module"
scope="prototype">
  <property name="displayName" value="sw#n" />
  <property name="name" value="swsharpn" />
  <property name="parameterContainer">
    <ref bean="swsharpnparams" />
  </property>
  <property name="path" value="swsharpn" />
</bean>
```

Ovakav način modeliranja modula i njihovih parametara pruža mogućnost iznimno jednostavnog dodavanja novih modula ili izmjenu postojećih. Za dodavanje novog modula dovoljno je napraviti novu instancu u konfiguracijskoj datoteci po uzoru na postojeće te prolagoditi njegove značajke (engl. *property*) da odgovaraju novom modulu.

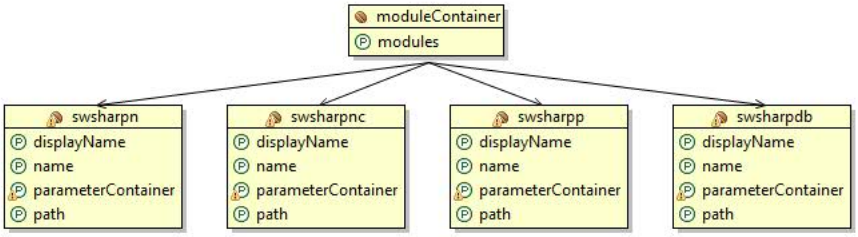
Izmjena postojećih modula je još jednostavnija. Dovoljno je promijeniti značajke postojeće instance unutar Spring konteksta na način da ona opisuje novonastalo stanje. Primjerice, ako skripta za pokretanje modula postane nedostupna kroz naredbu *swsharpn* i preseljena je u drugi direktorij (npr. /var), tada je dovoljno samo promijeniti vrijednost značajke *path* u */var/swsharpn*.

Nakon podešavanja modula dovoljno je dodati ih u *ModuleContainer*. To se također izvršava pomoću Spring konteksta na sljedeći način.

```
<bean id="moduleContainer"
class="hr.fer.zesoi.swsharp.modules.ModuleContainer"
scope="prototype">
  <property name="modules">
    <list>
      <ref bean="swsharpn" />
      <ref bean="swsharpnc" />
      <ref bean="swsharpp" />
      <ref bean="swsharpdb" />
    </list>
  </property>
</bean>
```

Dodavanje novog modula svedeno je na dodavanje nove reference na instancu modula u listu modula. Ta lista modula predaje se *ModuleContainer*-u kao argument metode za postavljanje vrijednosti nakon što se *ModuleContainer* instancira. Jednako tako moguće je onemogućiti pristup modulu putem web sučelja tako da se iz liste obriše referenca na taj modul.

Struktura instanci koja opisuje module i kontejner dana je na slici 6.8.



Slika 6.8: Prikaz strukture instanci razreda *Module* koji opisuju module

6.4. Paket hr.fer.zesoi.swsharp.databases

Paket hr.fer.zesoi.swsharp.databases sadrži dva razreda: Database i DatabaseContainer.

6.4.1. Razred Database

Razred *Database* služi kao opisnik baza podataka koje korisnik može odabrati pri izvršavanju modula SW#db, a koje su već pohranjene na čvrstom disku poslužitelja. To je jednostavan razred koji pohranjuje ime baze koje će se prikazati u formi za unos parametara za SW#db te putanju do same baze na čvrstom disku poslužitelja.

6.4.2. Razred DatabaseContainer

Razred *DatabaseContainer* služi kao kontejner za čuvanje baza podataka određenog modula (slično kao *ParameterContainer* za parametre).

6.4.3. Spring konfiguracija

Nova baza podataka dodaje se u web sučelje na sličan način kao i novi parametar ili novi modul. Potrebno je dodati novi instancu razreda *Database* u Spring kontekst na sljedeći način:

```
<bean id="pdbdb" class="hr.fer.zesoi.swsharp.databases.
    Database"
  scope="prototype">
  <property name="name" value="Protein Data Bank proteins
     " />
  <property name="path" value="/home/data/pdbaa.fasta" />
</bean>
```

Nakon toga se stvorena instanca razreda *Database* dodaje u *DatabaseContainer*.

```
<bean id="dbContainer" class="hr.fer.zesoi.swsharp.
    databases.DatabaseContainer"
  scope="prototype">
  <property name="databases">
    <list>
      <ref bean="pdbdb" />
```

```
    </list>
  </property>
</bean>
```

Potrebno je još samo postaviti stvoreni *DatabaseContainer* kao *databaseContainer* modula te omogućiti korištenje baza za modul na sljedeći način:

```
<bean id="swsharpdb" class="hr.fer.zesoi.swsharp.modules.
    Module"
  scope="prototype">
  ...
  <property name="usesDatabase">
    <value type="java.lang.Boolean">true</value>
  </property>
  <property name="databaseContainer">
    <ref bean="dbContainer" />
  </property>
  ...
</bean>
```

Ove promjene manifestirat će se u web formi za unos parametara modula na sljedeći način:

- u formu je dodan parametar sa labelom ”Choose database”,
- u padajućem izborniku prvotno je označena baza ”None”, te je
- u padajući izbornik dodana je baza opisana konfiguracijom za instancu *pdbdb*.

6.5. Paket hr.fer.zesoi.swsharp.controllers

Paket *hr.fer.zesoi.swsharp.controllers* sadrži četiri kontrolera čije metode pružaju ulazne točke korisnika u samo web sučelje.

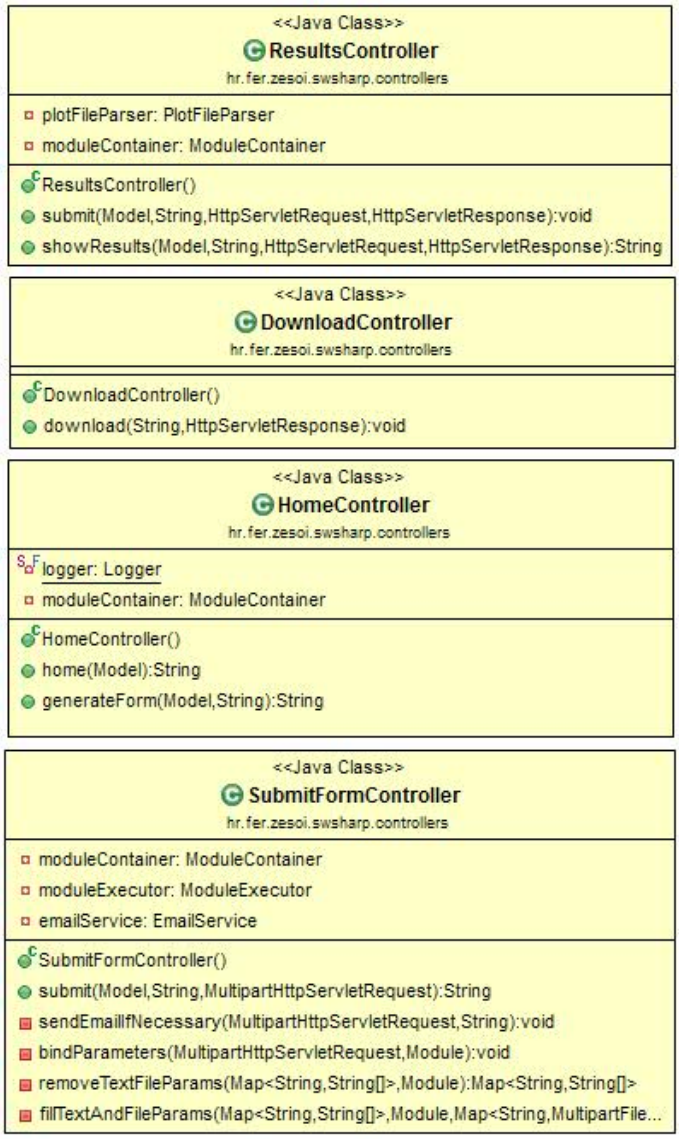
6.5.1. Web adrese

U daljnjem tekstu web adrese će se nazivati URL (engl. *Unified Resource Locator*). U njima će se koristiti nekoliko ključnih riječi koje redom označavaju:

host: IP adresa poslužitelja na kojeg je postavljeno web sučelje,

port: port na kojem je omogućen pristup web sučelju,

root: ime pod kojim je web sučelje postavljeno na poslužitelj, tj. *context-root* za web sučelje zadan u konfiguracijskoj datoteci poslužitelja web.xml



Slika 6.9: Class dijagram paketa hr.fer.zesoi.swsharp.controllers

6.5.2. HomeController

HomeController je kontroler kojem se pristupa pomoću dva URL-a:

1. http://*host*:*port*/*root*/
2. http://*host*:*port*/*root*/*imeModula*.

Ime modula označava modul za koji se prikazuje forma za unos parametara. Ako se pristupi URL-u pod 1., prikazuje se početna stranica s uputama za korištenje web

sučelja. Metode mapirane na spomenute URL-ove pune model potrebnim podacima za prikaz forme te ga prosljeđuju pogledu *home* ili *error* u slučaju da upisano *imeModula* ne postoji.

6.5.3. SubmitFormController

SubmitFormController je najvažniji kontroler za rad web sučelja. Njegova zadaća je obraditi parametre primljene u zahtjevu korisnika te proslijediti posao *ModuleExecutor*-u koji će zatim odraditi samo pokretanje modula u sklopu operacijskog sustava poslužitelja. Ovaj kontroler prima POST zahtjeve na URL-u:

1. http://*host*:*port*/*root*/*imeModula*/submit

Ime modula označava modul za koji je predana forma za unos parametara. Zahtjev se šalje nakon što korisnik klikne na gumb *Submit*, a njegov tip sadržaja (engl. *content-type*) je *multipart/form-data*. Takav tip zahtjeva omogućava korisniku postavljanje datoteke kao parametra za pokretanje SW# modula što se koristi za unos sljedova čije se poravnanje računa. Zahtjevi tog tipa prolaze kroz *CommonsMultipartResolver*. Taj razred je dio Spring biblioteke te omogućava jednostavniju manipulaciju parametrima zahtjeva te njihovo iščitavanje u obliku prigodnom za obradu.

Metoda od iznimne važnosti je

```
private void bindParameters(request, module)
```

Ona povezuje parametre primljene u zahtjevu *req* sa parametrima modula *module*. To je moguće iz razloga što pogled koji sadrži formu za unos parametara i ovaj kontroler imaju pristup istoj definiciji modula te imenima njegovih parametara. Takvu povezanost omogućena je krom XML konfiguracijsku datoteku Spring IoC kontejnera. Vezivanje parametara predanih kroz formu sa parametrima za pokretanje modula na poslužitelju vrši se prema sljedećem pseudokodu:

Važno je napomenuti da će metoda *fillTextAndFileParams* ,ako se kroz formu predaju i datoteka i ispuni tekstualno polje, kao parametar za pokretanje modula uzeti podatke iz datoteke te zanemariti one koji su uneseni kroz tekstualno polje.

Kontroler puni model podacima potrebnim za prikaz stranice koja se prikazuje kod čekanja rezultata. Ime tog pogleda je *progress*.

6.5.4. ResultsController

ResultsController je kontroler koji služi za prikaz rezultata poravnanja. Pristupa mu se kroz dva različita URL-a:

Algorithm 1 Povezivanje parametara forme i modula

fillTextAndFileParams() : povezivanje parametara tipa TextAndFile
removeTextAndFileParams(request) : povezivanje parametara tipa TextAndFile
requestParameters: parametri zahtjeva.
module: modul na koji treba mapirati parametre.
$moduleParameterContainer := module.getParameterContainer$
for $parameter : requestParameters$ **do**
  $parameterName := parameter.name$
  $moduleParameter := moduleParameterContainer.get(parameterName)$
  if $exists(moduleParameter)$ **then**
    $moduleParameter := parameter$: parametar pomocu svoje metode setValue zna kako da pohrani parametar predan u formi
  end if
end for

1. http://*host*:*port*/*root*/results/*jobId*/

2. http://*host*:*port*/*root*/results/*jobId*/show

Metoda mapirana na URL naveden pod 1. vrši provjeru da li je posao sa identifikatorom *jobId* završio. Ako je završio kao odgovor vraća pozivatelju odgovor s kodom 200 koji u terminologiji HTTP protokola označava OK. Ako posao nije završio, pozivatelju vraća odgovor s kodom 404 što znači da rezultati nisu pronađeni.

Metoda mapirana URL-om pod 2. provjerava da li je poravnanje s identifikatorom *jobId* gotovo te puni model podacima za prikaz rezultata. Taj model sastoji se od dva člana:

results: tekst iz datoteke nastale pokretanjem modula u formatu pairstat ili Blast m 9 ovisno o modulu

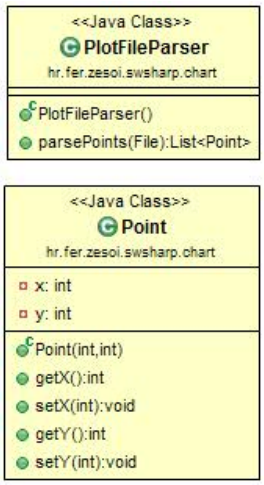
points: zapis točaka potrebnih za iscrtavanje grafičkog prikaza rezultata poravnanja.

Model se prosljeđuje pogledu *results*.

6.6. Paket hr.fer.zesoi.swsharp.chart

Razredi iz paketa *hr.fer.zesoi.swsharp.chart* služe za modeliranje podataka za grafički prikaz rezultata poravnanja. Razred *Point* opisuje točku u dvodimenzionalnom prostoru.

Razred *PlotFileParser* je razred koji čita iz datoteke koja sadrži rezultate zapisane u GNUplot formatu te vraća popis točaka iz datoteke u obliku liste objekata tipa *Point*.



Slika 6.10: Class dijagram paketa hr.fer.zesoi.swsharp.chart

7. Zaključak

U radu je predstavljeno web sučelje za module biblioteke za poravnanje sljedova SW#. Ukratko je objašnjena važnost poravnanja sljedova gledano sa stanovišta bioinformatike te je uvedena terminologija s područja problema poravnanja sljedova. Opisani su moduli biblioteke SW# te su pojašnjeni njihovi parametri. Iznesene su korisničke upute za korištenje web sučelja uz prikaz samog korisničkog sučelja. Opisane su najvažnije tehnologije i biblioteke korištene pri izradi ovog rada: Spring, JSTL, Maven i JavaScript. U opisu programskog ostvarenja opisani su Java paketi i razredi kojima je implementirano web sučelje te način na koji je konfiguracija web sučelja povezana s prikazom korisničkog sučelja web aplikacije. Pokazana je proširivost rješenja te su dane upute za moguće daljnje proširenje web sučelja novim modulima, parametrima modula te novim bazama podataka.

LITERATURA

[1] Commons wikimedia. URL http://commons.wikimedia.org/.

[2] Spring documentation. URL http://www.springsource.org/documentation.

[3] Richard C Deonier, Simon Tavaré, i Michael S Waterman. *Computational genome analysis: an introduction*. Springer, 2005.

[4] Steven Henikoff i Jorja G Henikoff. Amino acid substitution matrices from oprotein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[5] Matija Korpar. Sw - biblioteka za poravnanje sljedova korištenjem gračkih procesora. Magistarski rad, Fakultet elektrotehnike i računarstva, Zagreb, 2013.

[6] Gary Mak. *Spring Recipes: A Problem-Solution Approach (Books for Professionals by Professionals)*. Apress, 2008. ISBN 1590599799. URL http://www.amazon.de/Spring-Recipes-Problem-Solution-Approach-Professionals/dp/1590599799%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D1590599799.

[7] Douglas Crockford. *JavaScript: The Good Parts*. O'Reilly Media, Inc., 2008. ISBN 0596517742.

[8] David M. Geary. *Core JSTL: Mastering the JSP Standard Tag Library*. Prentice Hall Professional Technical Reference, 2002. ISBN 0131001531.

[9] Craig Walls i Ryan Breidenbach. *Spring in action*. Manning Publications Co., Greenwich, CT, USA, 2007. ISBN 9781933988139.

[10] Karl Swedberg i Jonathan Chaffer. *Learning jQuery: Better Interaction Design and Web Development with Simple JavaScript Techniques*. 2007. URL http://www.amazon.com/gp/product/1847192505/ref=reg_hu-wl_item-added/102-1745539-0800128.

[11] Bootstrap documentation. URL http://twitter.github.io/bootstrap/.

[12] Maven documentation. URL http://maven.apache.org/guides/.

[13] Flotjs project homepage. URL http://www.flotcharts.org/.

Web sučelje za poravnanje sljedova

Sažetak

Problem poravnanja sljedova jedan je od najvažnijih problema s područja bioinformatike, ali i ostalih područja znanosti. U ovom radu predstavljeno je web sučelje za poravnanje sljedova. Ono, kao svoju osnovu, koristi biblioteku SW# koja provodi poravnanje sljedova koristeći grafičke kartice zasnovane na CUDA arhitekturi. Važnost web sučelja leži u tome što se njime istraživačima s područja bioinformatike omogućuje iskorištavanje brzih i efikasnih modula biblioteke SW# bez potrebe za nabavkom sklopovske opreme koja je nužna za njihovo izvođenje. Prikazano web sučelje proširivo je novim modulima, parametrima te bazama proteina i nukleotida.

Ključne riječi: poravnanje sljedova, web sučelje, Java, SW#, Spring

Web interface for sequence alignment

Abstract

Sequence alignment is one of the most important problems in todays bioinformatics as well as in some other fields of science. Web interface for sequence alignment is introduced in this thesis. It uses SW# library as its basis. The SW# library utilizes CUDA enabled graphic processors for its execution. The importance of the web interface introduced lays in the fact that it enables researchers from the field of bioinformatics to use efficient and fast SW# module without the need to acquire hardware required to run SW# modules. The web inteface that has introduced is expendable with new modules, their parameters and protein and nucleotide databases.

Keywords: sequence alignment, web interface, Java, SW#, Spring