Projectgroep 10

Document	Ontwerp.docx
Datum	14-03-2018
Versie	1.0
Groepsleden	Damian Bolwerk
	Jonathan Feenstra
	Fini De Gruyter
	Lotte Houwen
	Alex Janse

ORFpred

Open Reading Frame predictie tool

Versiebeheer

Versie	Datum	Auteur	Beschrijving
0.1	07-03-2018	Fini + Jonathan + Alex	Systeemarchitectuur diagram + ERD model + Class diagram gemaakt.
0.2	08-03-2018	Fini	Beschrijving Systeemarchitectuur toegevoegd en methodes uitgeschreven, begrippenlijst uitgewerkt.
0.3	09-03-2018	Fini	Laatste bewerkingen systeemarchitectuur en begrippenlijst.
0.4	09-03-2018	Jonathan	Entiteiten toegelicht, gekozen design pattern uitgelegd en begrippen toegevoegd.
0.5	11-03-2018	Lotte	Inleiding geschreven.
0.6	12-03-2018	Alex	Class diagram aangepast.
0.7	13-03-2018	Jonathan + Fini	ERD verder uitgewerkt en verbeterd.
0.8	14-03-2018	Damian + Lotte	Project beschrijving en tekst bij software architectuur geschreven.
1.0	14-03-2018	Allen	Laatste aanpassingen en overleg.

Contents

1. Introductie

Aanleiding

Een reading frame is een DNA-sequentie opgedeeld in codons vanaf een bepaalde positie. Aangezien elk codon bestaat uit drie nucleotiden en DNA zowel van voren als naar achter kan worden gelezen, zijn er in totaal zes mogelijke reading frames. In een reading frame wordt vaak gezocht naar open reading frames (ORF's).

Een ORF kan in de moleculaire biologie een handig hulpmiddel zijn om meer te weten over de genen van een bepaald organisme (Brent, 2005). Als deze ORF's ook geannoteerd kunnen worden, kan dit veel informatie verschaffen over bijvoorbeeld de identiteit (soort) en functionaliteit (in vergelijking met soortgenoten) van het organisme (Stein, 2001). Daarnaast kunnen ORF's gebruikt worden in experimenteel onderzoek naar gen functies, zoals knock-out experimenten en het meten van genexpressies onder verschillende omstandigheden (Pevsner, 2015). Tenslotte is het mogelijk om nieuwe genen te vinden door ORF's in een genoom te voorspellen, die mogelijk coderend zijn (Stein, 2001). Aangezien het voorspellen en annoteren van ORF's in een genoom handmatig veel te veel tijd zou kosten, is het van belang om dit geautomatiseerd te kunnen doen.

Doel van dit document

In dit document is in detail weergeven waaruit de applicatie bestaat en waaruit het is opgebouwd. In dit document zullen zal er tevens verantwoording worden afgelegd voor gemaakte keuzes qua ontwerp en tools. Er is onder andere een uitgebreide beschrijving te vinden van de systeemarchitectuur. Deze geeft de eindgebruiker en de ontwikkelaars meer inzicht in de functionaliteit.

Doel van de applicatie

Handmatig zoeken naar ORF's in een DNA-sequentie is een ondoenlijke taak. Door gebruik te maken van bio-informatica tools zoals de Basic Local Alignment Search Tool (BLAST) (Altschul et al., 1990) kan het proces meer geautomatiseerd worden. Het doel van dit project is om een applicatie te ontwikkelen die de ORF's kan voorspellen op basis van een ingevoerde DNA-sequentie. De applicatie zal de DNA-sequentie opslaan in een database. Van de DNA-sequentie worden alle mogelijke ORF's voorspeld en eventueel opgeslagen in een database. De applicatie houdt hierbij rekening met de diverse reading frames voor de te voorspellen ORF's. Deze verschillende reading frames worden ook als aminozuursequenties gevisualiseerd in de applicatie. Per ORF zal er een BLAST-search tool beschikbaar zijn. Deze resultaten worden gevisualiseerd in de applicatie en opgeslagen in de database.

Doelgroep

Dit document is bedoeld voor de ontwikkelaars van de applicatie. Het document zal een overzicht geven van alle eisen waaraan de applicatie moet voldoen en hoe de applicatie is opgezet. Het kan zodoende gebruikt worden bij het ontwerpen van de applicatie.

2. Project beschrijving

Startpunt van het project

De applicatie is verder gebouwd op de bestaande module van BioJava (Prlić et al., 2012) en van NCBI API (Coordinators 2016). De BioJava module heeft als functionaliteit om bestanden te parsen en om gemakkelijk bepaalde handelingen uit te voeren op biologische sequenties, zoals de transcriptie van een DNA-sequentie. In deze applicatie moet er gecommuniceerd worden met de NCBI-server om BLAST (Altschul et al., 1990) searches uit te voeren, dit wordt gedaan met behulp van NCBI API (Coordinators 2016) waarmee de gewenste parameters meegegeven kunnen worden en vervolgens de resultaten geretourneerd kunnen worden. Een mogelijke restrictie bij het opschalen van dit project zou kunnen zijn dat de NCBI-server slechts een beperkte hoeveelheid BLAST searches tegelijk kan verwerken, in dat geval is het raadzaam om in plaats hiervan lokaal BLAST searches uit te voeren.

Gekozen technieken

Alle modules van onze applicatie zijn in Java geschreven zodat de applicatie werkzaam is op elk platform. Binnen dit project is echter gekozen om alleen garanties te bieden voor bepaalde versies van Windows en Linux, maar zou in principe ook werken op andere versies of besturingssystemen. De applicatie is gemaakt binnen een Maven project (Miller et al., 2010) om het Java project te beheren. De applicatie maakt gebruikt van een SQL-database die gevuld wordt met behulp van SQL queries (SQL Developer, Versie 17.4). De database die door de applicatie gevuld wordt, heeft nog geen definitieve locatie gekregen. Om de applicatielogica te scheiden van de user interface, zodat code efficiënt kan worden hergebruikt, is er gekozen voor het model-view-controller (MVC) design pattern. Er wordt met de NCBI-server gecommuniceerd met het communicatieprotocol HTTP- Hypertext Transfer Protocol. SQL maakt gebruikt van de communicatieprotocollen TCP - Transmission Control Protocol en UDP - User Datagram Protocol.

Design principles

De applicatie is aan de hand van het model-view-controller (MVC) design pattern geschreven waardoor alleen de front-end tier van veranderd hoeft te worden om er bijvoorbeeld een webbased applicatie van te maken. Er kan gekozen worden om de database waar de gegenereerde data wordt opgeslagen lokaal te maken zodat er meer BLAST searches tegelijkertijd kunnen worden uitgevoerd. Hierdoor kan de applicatie beter opgeschaald worden.

Design patterns

Om de applicatielogica te scheiden van de user interface zodat code efficiënt kan worden hergebruikt, is er gekozen voor het model-view-controller (MVC) design pattern. Hierbij gebruikt de gebruiker de controller (de event handler die de gebruikers input omzet naar opdrachten) om het model (de data, logica en regels van de applicatie) te manipuleren, die vervolgens de view (de GUI) updatet. Voordelen van het MVC design pattern zijn dat de developers in parallel aan verschillende componenten van de applicatie kunnen werken, zonder elkaar in de weg te staan. Bovendien kan de code gemakkelijk hergebruikt worden voor andere applicaties door alleen de front-end aan te passen (Reenskaug & Coplien, 2009). In het geval van ORFpred is de gebruiker de bioloog, de controller de GUIEventHandler (inner class van GUI), het model bestaat uit meerdere classes en de view is de GUI.

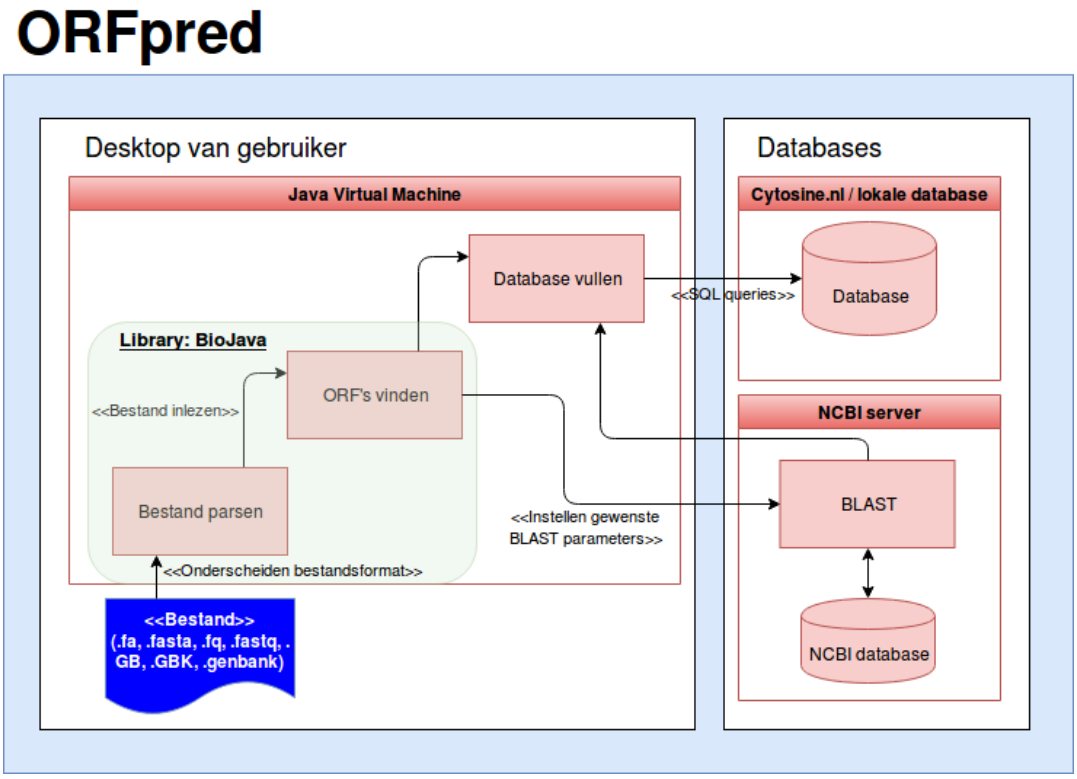
3. Systeemarchitectuur

Doel van de systeemarchitectuur

Het doel van de systeemarchitectuur is om globaal te laten zien hoe de architectuur die het uiteindelijke systeem krijgt is opgebouwd. De bedoeling is om de eindgebruiker en de ontwikkelaars hierdoor meer inzicht te geven in de te bouwen functionaliteit. Bovendien beschrijft de systeemarchitectuur ook zaken als gebruikte software en modules, waardoor de requirements van het uiteindelijke systeem worden verduidelijkt. In dit systeemarchitectuur is voornamelijk de middle- en back-end tier te zien.

Gedetailleerde systeemarchitectuur

De systeemarchitectuur is gevisualiseerd om een duidelijker beeld te krijgen van de opbouw van het systeem (figuur 1). De scope of omvang van de applicatie is dat iedereen met een DNA sequentie in staat is om ORF's te voorspellen en er een functie aan te verbinden met behulp van vergelijkende genoomanalyse. Hierdoor kan de gebruiker snel meer te weten komen over het mogelijke coderend materiaal in de sequentie. Deze applicatie zal in staat zijn om zowel op Windows als Linux besturingssystemen te draaien (zie systeemspecificaties).



Figuur 1: Het gedetailleerde systeemarchitectuur van de ORFpred applicatie. Hierin zijn twee deelsystemen getoond, namelijk de 'desktop van gebruiker' en 'databases'. Binnen deze twee deelsystemen zijn vier modules verwerkt, namelijk 'Bestand parsen', 'ORF's vinden', 'Database vullen' en 'BLAST'. Er zijn twee databases, waarbij de NCBI database een externe database is waar informatie uit opgehaald wordt en de lokale/cytosine database door de applicatie gebruikt wordt om de gegenereerde data in op te slaan. De applicatie heeft een bestand met DNA sequentie(s) als input met een biologisch fileformat. De meeste modules draaien in de Java Virtual Machine, waarin de Java applicatie kan draaien. De databases staan op externe servers en de module 'BLAST' wordt ook uitgevoerd op de externe NCBI database. Figuur is gemaakt met behulp van Draw.io, versie 8.2.3

De systeemarchitectuur bestaat uit twee deelsystemen, namelijk de desktop van de gebruiker en de databases (figuur 1). De kern van de functionaliteit vindt plaats op de desktop en draait op een Java Virtual machine (JVM) (Versienummers van deze en andere software, zie "Systeemspecificaties"). Er zijn vier modules, waarvan drie uitgevoerd worden op het eerste deelsysteem.

Allereerst kan de gebruiker een bestand kiezen met een sequentie. Ondersteunde biologische bestandsformaten zijn hierbij .fa, .fasta, .fq, .fastq, .GBK, .GB en .genbank. Deze wordt vervolgens door de eerste module verwerkt, genaamd "bestand parsen" (beschrijving van deze en andere modules, zie hieronder). Het bestand kan vervolgens door de applicatie worden ingelezen, waarna de volgende module "ORF's vinden" kan functioneren. Hierna zijn er twee routes mogelijk.

Allereerst kunnen de gevonden ORF's worden opgeslagen in de database met de module "Database vullen". Daarnaast kunnen de gevonden ORF's eerst geannoteerd worden met de module "BLAST", om vervolgens alsnog met de module "Database vullen" opgeslagen te worden in de database. De gebruiker geeft hierbij aan welke ORF's geBLAST worden en met welke parameters. De database voor het opslaan van de data is een SQL database die met behulp van SQL queries gevuld moet worden (SQL Developer, Versie 17.4). De module "BLAST" vindt plaats op de NCBI server, waarbij de juiste data opgehaald wordt uit de NCBI database uit de NCBI server met behulp van de NCBI API (Coordinators 2016). De database waar de resultaten van deze applicatie via de module "Database vullen" gevuld wordt, heeft nog geen definitieve locatie gekregen. Het is op dit moment nog niet duidelijk of de cytosine.nl server hiervoor beschikbaar is. Een alternatief is het bouwen van een lokale database, maar heeft als nadeel dat deze alleen te benaderen is op de computer waar de database gemaakt is.

Bij twee modules wordt er gebruik gemaakt van bestaande Java libraries voor de functionaliteit van de applicatie, namelijk BioJava (Prlić et al., 2012) en de NCBI API (Coordinators 2016). Deze libraries worden gebruikt, omdat bepaalde bewerkingen al door anderen zijn uitgewerkt. Concrete voorbeelden zijn het parsen van het bestand, translatie van DNA naar RNA naar aminozuren/stopcodons en het geautomatiseerd gebruiken van de NCBI BLAST. BioJava is in staat om meerdere bestandsformaten in te lezen en bepaalde informatie hieruit te destilleren, zoals de DNA sequentie, en deze vervolgens te gebruiken om te vertalen naar RNA en aminozuren. De NCBI API kan geautomatiseerd sequenties BLASTen met de gewenste parameters en vervolgens de resultaten hiervan retourneren.

Samenvattend is ORFpred in staat om met de juiste bestandsformaten als input, ORF's te voorspellen, deze optioneel te annoteren met vergelijkende genoomanalyse en vervolgens op te slaan in een SQL database. Deze data kan vervolgens op elk gewenst moment aangeroepen worden door de gebruiker om de data te bekijken. De gebruikte modules hiervoor zijn:

Bestand parsen

Deze module zorgt voor de juiste verwerking van het bestand, door deze om te zetten naar een sequentie. Het kan uit GenBank, FASTA en FASTQ bestanden de juiste informatie (sequentie, header ect.) destilleren. De gebruikte taal is Java met de library BioJava (Prlić et al., 2012). Als protocol heeft deze module het onderscheid kunnen maken tussen de verschillende bestandsformaten.

ORF's vinden

Deze module kan uit een DNA-sequentie naar een aminozuursequentie transleren en daar vervolgens ORF's in vinden door te kijken naar de stopcodons. Een ORF wordt gevonden als de ruimte tussen twee stopcodons in een bepaalde reading frame meer is dan de minimale ORF lengte opgegeven door de gebruiker. De gebruikte taal is Java en de translatie van DNA naar aminozuren vindt plaats met behulp van de library BioJava (Prlić et al., 2012).

Database vullen

De database vullen module vult de database met ORF data, waardoor de data wordt opgeslagen en later weer bekeken kan worden. Optioneel kan, als er geBLAST is, ook BLAST gegevens in de database worden opgeslagen. De gebruikte taal is Java, waarbij SQL queries gebruikt worden om data toe te voegen aan de SQL database (SQL Developer, Versie 17.4).

BLAST

Deze BLAST module kan de gevonden ORF(s) via de server van het NCBI BLASTen, bedoeld voor de functie-annotatie van de betreffende sequenties (Altschul et al., 1990, Coordinators 2016). Dit is nodig om te bepalen of de ORF mogelijk een gedeelte van een coderend stuk is. De gebruikte taal is Java, waarbij gebruik gemaakt wordt van de NCBI API (Coordinators 2016) om geautomatiseerd te kunnen BLASTen met de juiste parameters.

Systeemspecificaties

ORFpred	**<<Desktop van gebruiker>>** - Apparaat uitvoer programma		**<<Software>>***	**<<Versienr. (minimum)>>**
			Linux (OS)	Ubuntu 16.04 LTS
			Windows (OS)	Windows 7
		Module 1 **<<Bestand parsen>>**	Java, BioJava (Java Library), Maven (Java project tool)	Java: "1.8.0_161" JRE: "build 1.8.0_161-b12" JVM: "1.8" Maven: "3.5.1" BioJava: "4.2.8"
		Module 2 **<<ORF's vinden>>**	Java, BioJava (Java Library), Maven (Java project tool)	Java: "1.8.0_161" JRE: "build 1.8.0_161-b12" JVM: "1.8" Maven: "3.5.1" BioJava: "4.2.8"
		Module 3 **<<Database vullen>>**	Java, SQL developer, Maven (Java project tool)	Java: "1.8.0_161" JRE: "build 1.8.0_161-b12" JVM: "1.8" Maven: "3.5.1" SQL Developer: "17.4"
	<<Databases>> - Cytosine.nl (HAN servers) of Apparaat uitvoer programma (lokaal) - BLAST (NCBI servers)			
		Module 4 **<<BLAST>>**	Java, Maven (Java project tool), BLAST API	Java: "1.8.0_161" JRE: "build 1.8.0_161-b12" JVM: "1.8" Maven: "3.5.1" BLAST+: "2.6.0"

*Referenties: (Altschul et al., 1990, Coordinators 2016, Miller et al., 2010, Prlić et al., 2012, SQL Developer, Versie 17.4)

De applicatie is gemaakt in een Maven project, binnen de programmeertaal Java. Maven is een tool die gebruikt wordt om Java projecten te beheren (Miller et al., 2010). Alle Java applicaties werken binnen een Java virtual machine (JVM), zodat de applicatie in principe kan werken op elk besturingssysteem (OS) dat de juiste JVM kan draaien. Binnen dit project is echter gekozen om alleen garanties te bieden voor bepaalde versies van Windows en Linux, maar zou dus ook kunnen werken op andere of oudere besturingssystemen.

Modules 1 en 2 maken gebruik van de BioJava library (Prlić et al., 2012). Deze library kan voor veel biologische toepassingen worden gebruikt en is een handige tool om bepaalde biologische berekeningen/bewerkingen die in de biologie veel voorkomen, zoals DNA translatie, gemakkelijk uit te voeren. Module 4 maakt gebruik van de BLAST library (Coordinators 2016). Deze library maakt het mogelijk om geautomatiseerd gebruik te maken van de NCBI database, zodat dit veel minder tijdrovend wordt, dan als deze bewerkingen met de hand uitgevoerd zouden moeten worden. Module 3 maakt gebruik van SQL Developer (SQL Developer, Versie 17.4) om de SQL database te vullen met SQL queries.

4. Software-architectuur

Doel van de software-architectuur

Met de software-architectuur zal er een overzicht worden gegeven over de verschillende classes en enumerations (en packages) waaruit de te maken software is opgebouwd. Hierdoor is de ontwikkelaar in staat om te bepalen welke classes er geschreven moeten worden en wat de relatie is hiertussen. Bovendien toont dit de methodes die elke class in ieder geval moet bevatten.

UML Class Diagram(s)

ReadingFramer

Deze class accepteert een DNA-sequentie en geeft daarbij de 6 reading frames als eiwitsequenties terug. Zo kunnen er in elke mogelijke reading frame ORF's worden gezocht.

ORFFinder

Deze class zoekt in de gegeven eiwitsequentie via een regular expression naar ORF's die voldoen aan de ingestelde minimum lengte, en slaat deze op in een ArrayList.

ORFHighlighter

Deze class zorgt ervoor dat de voorspelde ORF's in de gegeven sequentie met de ingestelde kleur gemarkeerd worden. Hierdoor kan de gebruiker ze gemakkelijk in de gevisualiseerde reading frames herkennen.

GUI

De GUI class zorgt ervoor dat de gebruiker interactie kan aangaan met de functionaliteiten van de applicatie via een grafische interface. De GUI class geeft deze interacties door aan de GUIUpdater class via event listeners.

GUIUpdater

Deze class ontvangt de aangeroepen functionaliteit, roept hierbij de juiste methoden aan en zorgt ervoor dat de resultaten worden weergegeven in de GUI.

FileHandler

De FileHandler zorgt ervoor dat er een lokaal bestand kan worden geopend, waarvan het bestandstype wordt bepaald via de FileType enumerator. Dit bestand wordt vervolgens gebruikt om de headers met bijbehorende sequentie op te slaan en te retourneren naar de GUIUpdater class.

DatabaseConnector

Zorgt ervoor dat er connectie gemaakt kan worden met de SQL database waarin de relevante gegevens kunnen worden opgeslagen en uit kunnen worden opgehaald.

DatabaseLoader

Laadt de geselecteerde gegevens van de database in met behulp van SQL queries.

DatabaseSaver

Slaat gegevens op in de database via SQL queries.

BLASTer

BLAST een query met de geselecteerde instellingen tegen een gekozen database.

BLASTResult

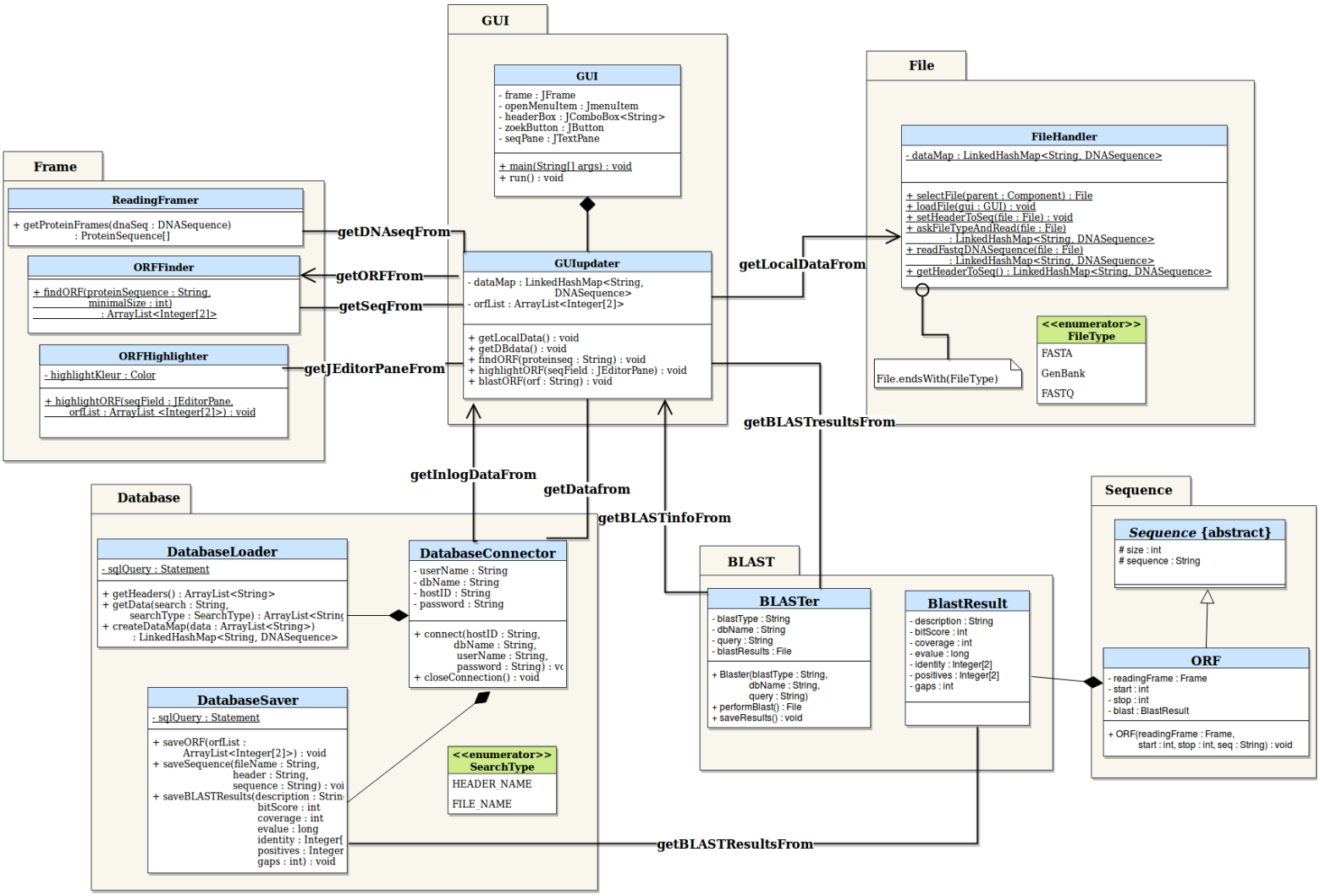
Slaat de benodigde informatie van de BLAST op.

Sequence

Abstracte class die de attributen van een biologische sequentie bevat. Deze wordt gebruikt voor overerving.

ORF

Erft over uit de Sequence class en wordt gebruikt om informatie per ORF op te slaan, bijvoorbeeld de start- en stop positie binnen een reading frame.



Figuur 2. Een class diagram van de applicatie, waarin is te zien welke methodes en eigenschappen de verschillende classes bevatten. De classes zijn onderverdeeld in verschillende packages, met ieder zijn eigen functionaliteit binnen de applicatie, zoals BLASTen, opslag van data en de algehele functionaliteit van de applicatie in de vorm van een GUI. De packages hebben onderling associaties met de desbetreffende classes. Deze class diagram is gemaakt met behulp van draw.io, versie 8.2.3.

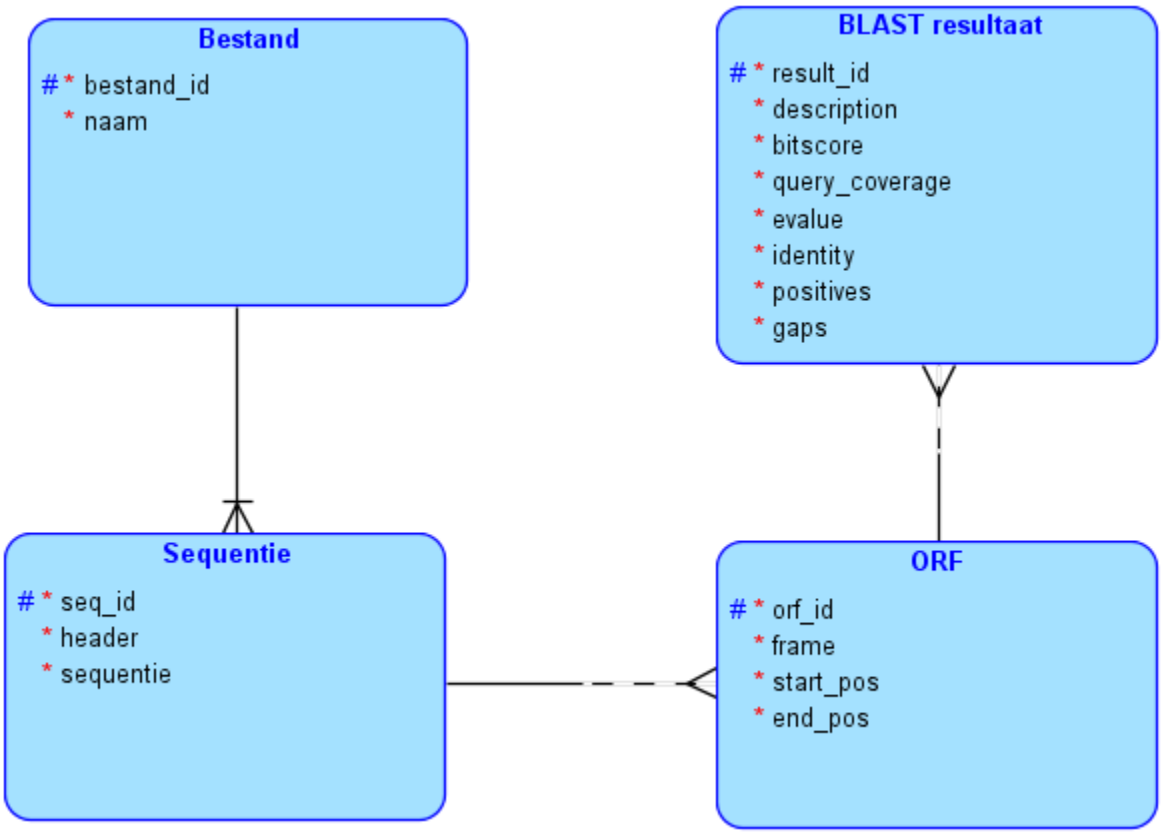
5. Technische gegevens structuur

Doel van de technische gegevens structuur

Het doel van de technische gegevens structuur is om de gegevensopslag in een rationeel model vast te leggen in de vorm van een Entity Relationship Diagram (ERD). Deze wordt gebruikt als ontwerp voor de relationele database waarin alle relevante gegevens worden opgeslagen.

Logisch model

Het logisch model van de database is een logisch ERD die een schematische weergave biedt van entiteiten, attributen en relaties. Dit ERD is genormaliseerd tot in de derde normaalvorm en weergegeven in figuur 3.



Figuur 3. Logisch ERD van de ORFpred database. Primary keys worden voorafgegaan met een blauw hekje (#), verplichte attributen met een rode asterisk (*). De relaties tussen entiteiten zijn in crow's foot notation weergegeven. Dit ERD is gemaakt met Oracle SQL Developer Data Modeler Versie 17.4.

Bestand

Sequenties worden per ingeladen bestand opgeslagen. Het bestandstype is af te leiden uit de extensie in de bestandsnaam en wordt daarom niet apart opgeslagen.

Sequentie

Omdat headers niet per definitie uniek zijn, wordt gebruik gemaakt van een aparte identifier als primary key. In het geval van GenBank bestanden wordt de (eerste) accessiecode als header gebruikt.

ORF

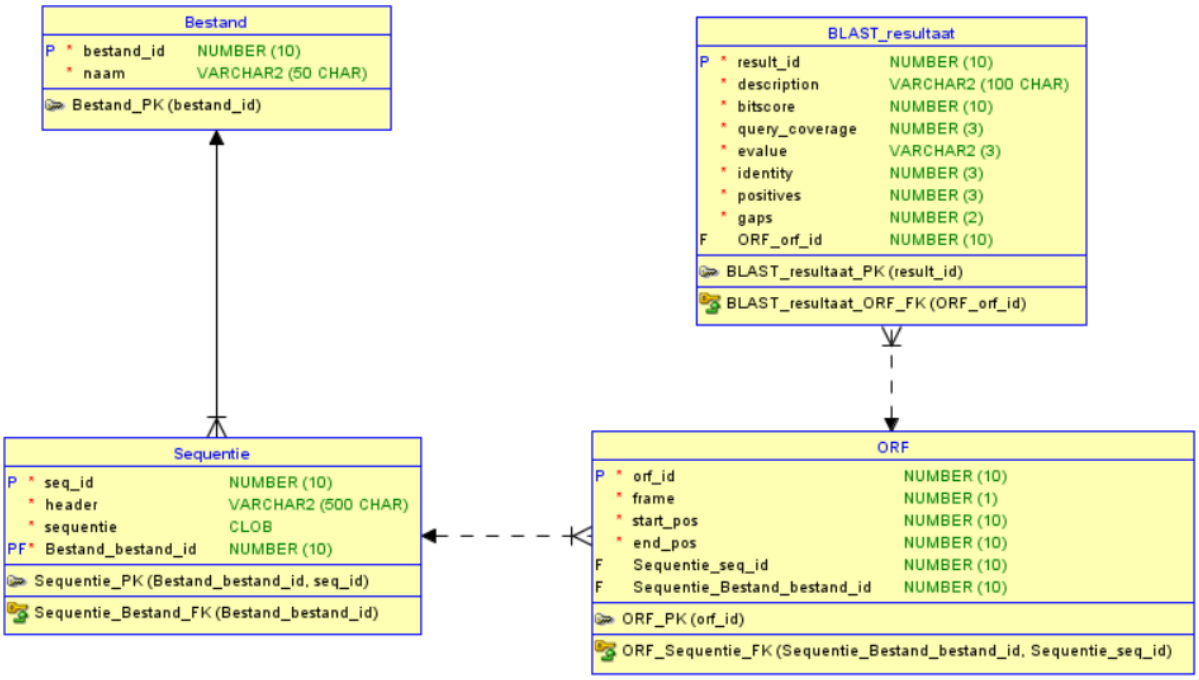
De ORF sequentie is af te leiden uit de start en eind posities in combinatie met de volledige sequentie. Het attribuut 'frame' heeft een CHECK constraint om de integriteit van opgeslagen gegevens te garanderen: "`frame BETWEEN 1 AND 6`" (niet in het ERD weergegeven).

BLAST resultaat

De accessiecodes staan (meestal) in de description en worden daarom niet apart opgeslagen. Het attribuut 'evalue' accepteert ook wetenschappelijke notatie, zoals '1e-7' en slaat het daarom op als VARCHAR, deze kunnen wanneer nodig geconverteerd worden naar getallen.

Technisch model

In figuur 4 is het technische ERD weergegeven waarin ook foreign keys, constraints en datatypen zijn weergegeven.



Figuur 4. Technisch ERD van de ORFpred database. Datatypen zijn in het groen achter de attributen weergegeven met tussen haakjes de maximale grootte waar van toepassing. De relaties tussen entiteiten zijn in crow's foot notation weergegeven. CLOB (Character Large OBject) is een datatype waarin een grote hoeveelheid karakters kunnen worden opgeslagen. Dit ERD is gemaakt met Oracle SQL Developer Data Modeler Versie 17.4.

6. Begrippenlijst

Begrip	Betekenis
ORF	Staat voor: Open Reading Frame. Een van de 6 mogelijke leesramen waarin een mRNA mogelijk vertaald kan worden in een eiwit.
BLAST	Staat voor: Basic Local Alignment Search Tool. Een algoritme om nucleotide of eiwit sequenties met een sequentie database te vergelijken en de statistische significantie te berekenen (Altschul et al., 1990).
OS	Staat voor: Operating system. Dit is het besturingssysteem waar de computer op draait, zoals MacOS, Linux Ubuntu, Windows 10 ect.
JVM	Staat voor: Java Virtual Machine. Een platformonafhankelijke omgeving voor het uitvoeren van Java bytecode.
MVC	Staat voor: Model-View-Controller. Een design pattern waarbij de applicatie logica gescheiden van de user interfaces.
FASTA	Bestandsformaat met headers beginnend met '>' en daaronder de sequentie. FASTA slaat de biologische sequentie op en in de header wordt informatie over de betreffende sequentie opgeslagen
FASTQ	Bestandsformaat met headers beginnend met '@', daaronder de sequentie, daaronder een '+' en vervolgens de kwaliteitsscore in ASCII formaat. FASTQ slaat zowel de biologische sequentie als de corresponderende kwaliteitsscores op van elke nucleotide.
GenBank	Bestandsformaat van de NIH genetic sequence database, dat behalve de sequenties ook annotatie bevat. Er zijn vaste woorden aan het begin van een regel die bepaalde informatie weergeeft. Elke entry begint altijd met het woord 'LOCUS', de sequentie staat altijd achter 'ORIGIN' en het einde van de entry wordt aangegeven met '//'.
LTS	Staat voor: Long-Time-Support. Dit zijn Linux OS versies die lang ondersteund wordt in werking van voornamelijk software.
JRE	staat voor: Java Runtime Environment. Is een runtime-omgeving die het mogelijk maakt om Java-programma's op een computer uit te voeren.
Maven	Een tool die gebruikt kan worden om Java projecten te beheren (Miller et al., 2010).
Regular expression.	Een zoekpatroon bestaande uit een sequentie van karakters.
ERD	Staat voor: Entity Relationship Diagram. Grafische representatie van entiteiten en hun relaties met elkaar, om relationele databases te modelleren.
Constraint	Eis waaraan een waarde moet voldoen om in een kolom opgeslagen te kunnen worden.
Primary key	Unieke identifier voor rijen in een databasetabel.
Foreign key	Unieke identifier voor rijen uit een gerelateerde databasetabel.
Crow’s foot notation	Wijze waarop in een ERD relaties tussen entiteiten kunnen worden aangeduid.
Communicatieprotocol	Bij communicatie in een computernetwerk wordt informatie uitgewisseld tussen de verschillende netwerkcomponenten. Dat gebeurt dan via een bepaalde standaard, een protocol. Meestal wordt zo’n protocol beschreven door aan te geven hoe elke afzonderlijke component in het netwerk zich moet gedragen.

7. Bronvermelding

- Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. (1990). Basic local alignment search tool. *J. Mol. Biol.* 215:403-410.
- Brent, M. R. (2005). Genome annotation past, present, and future: how to define an ORF at each locus. *Genome research*, *15*(12), 1777-1786.
- Coordinators, N. R. (2016). Database resources of the national center for biotechnology information. *Nucleic acids research*, *44*(Database issue), D7.
- Draw.io (Versie 8.2.3). Desktop diagramming application by the technology leaders in web diagramming. Northampton UK: Atlassian Confluence.
- Miller, F.P., Vandome, A.F. & McBrewster J. (2010). *Apache Maven*. Alpha Press.
- Oracle SQL Developer Data Modeler (Versie 17.4) [Computerprogramma]. Redwood City: Oracle Corporation.
- Pevsner, J. (2015). *Bioinformatics and Functional Genomics third edition.* John Wiley & Sons, Inc.
- Prlić, A., Yates, A., Bliven, S. E., Rose, P. W., Jacobsen, J., Troshin, P. V., ... & Holland, R. (2012). BioJava: an open-source framework for bioinformatics in 2012. *Bioinformatics*, *28*(20), 2693-2695.
- Reenskaug, J. & Coplien, J.O. (2009, 20 maart). *The DCI Architecture: A New Vision of Object-Oriented Programming*. Geraadpleegd op 9 maart 2018, van https://www.artima.com/articles/dci_vision.html
- Stein, L. (2001). Genome annotation: from sequence to biology. *Nature reviews genetics*, *2*(7), 493.