Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Jan-Christoph Meier

Entwicklung einer Webplattform zur Analyse und Visualisierung von molekularbiologischen Daten im Kontext der Multiple Sklerose Forschung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jan-Christoph Meier

Entwicklung einer Webplattform zur Analyse und Visualisierung von molekularbiologischen Daten im Kontext der Multiple Sklerose Forschung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Olaf Zukunft
Zweitgutachter : Prof. Dr. Stefan Sarstedt

Abgegeben am 25. August 2011

Jan-Christoph Meier

Thema der Bachelorarbeit

Entwicklung einer Webplattform zur Analyse und Visualisierung von molekularbiologischen Daten im Kontext der Multiple Sklerose Forschung

Stichworte

T-Zell-Aktivierung, MHC-II Molekül Bindung, Django-Webframework, Multiple Sklerose, HLA-Type, Bindungsmotiv

Kurzzusammenfassung

Gegenstand der Arbeit ist die Entwicklung einer Webplattform zur Analyse von Proteinsequenzen im Kontext der Multiple Sklerose Forschung. Hierfür werden zwei Programme der Bioinformatik, SYFPEITHI und das »MHC-II binding prediction«-Programm gekoppelt. Die Funktionalität der beiden Programme wird durch die Integration in eine eigene Software erweitert. Die Analyseergebnisse werden dem Benutzer auf unterschiedliche Arten visualisiert und in einer Datenbank abgespeichert. Entwickelt wird die Software in der Programmiersprache Python mit dem Django-Webframework unter Verwendung der Web-Technologie jQuery.

Jan-Christoph Meier

Title of the paper

Development of a web application to analyse and visualise molecular biology data for the multiple sclerosis research

Keywords

T-Cell activation, MHC-II binding prediction, Django Webframework, Multiple Sclerosis, HLA-Type, binding motif

Abstract

Topic of this paper is the development of a web application which combines two tools for the binding prediction of peptides on MHC-II molecules. The new developed software enhances and visualises the output of the two bioinformatics applications. The new gathered scientific data is stored in a relational database and is used for further analysis. The software is developed as a rich internet application and makes use of the Django webframework and advanced web technologies like the jQuery JavaScript library.

Danksagung

Dem Direktor des Inims-Institut Prof. Dr. Roland Martin für die Grundidee und die Ermöglichung dieser Arbeit.

Malte Mohme für die vielen wissenschaftlichen Anregungen und die Unterstützung in allen Fragestellungen der Immunologie- und MS-Forschung.

Dr. Hans-Martin Ziethen als Leiter der IT-Servicegruppe des ZMNH für die Beratung und Ideengebung aus Sicht der Informatik und die Bereitstellung der IT-Infrastruktur.

Prof. Dr. Olaf Zukunft für die Betreuung der Arbeit.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Das Institut für Neuroimmunologie und klinische Multiple Sklerose Forschung (Inims) am Zentrum für molekulare Neurobiologie Hamburg (ZMNH) hat sich zum Ziel gesetzt, die Ursache der Mutliple Sklerose-Erkrankung zu erforschen und mögliche Therapien zu entwickeln. Multiple Sklerose ist eine Autoimmunkrankheit, bei der das Immunsystem den eigenen Körper angreift. Die Ursachen dafür sind bis heute nicht endgültig erforscht.
Bei einer Multiple Sklerose Erkrankung werden körpereigene Zellen als fremd erkannt und durch das Immunsystem bekämpft. Ein Bestandteil des Immunsystems sind die T-Zellen. Werden sie angeregt, wird eine Immunreaktion ausgelöst. Damit es zu einer Immunreaktion kommt, müssen körperfremde Proteine den T-Zellen präsentiert werden, dies geschieht durch den Haupthistokompatibilitätskomplex (Abk. MHC von engl. Major Histocompatibility Complex). Die im menschlichen Körper vorhandenen MHC-Moleküle werden als Human Leukocyte Antigen (kurz HLA) bezeichnet. Zellen enthalten Proteine, bei denen es sich um zusammenhängende Ketten von Aminosäuren handelt. Damit der MHC-Komplex einen Fremdkörper, d.h. ein Protein, der T-Zelle präsentiert, muss das Protein an den MHC-Komplex binden. Damit die Bindung zustande kommt, muss das Protein zerkleinert werden, dieser Vorgang wird als MHC-Restriktion bezeichnet.
Ein Ansatz in der Multiple Sklerose Forschung ist die Fragestellung, warum die T-Zelle körpereigene Proteine als fremd erkennt und eine Immunreaktion auslöst. Hierzu wird eine möglichst hohe Anzahl viraler Proteine auf die Bindung an dem MHC-Komplex untersucht. Bindet das Protein an den MHC-Komplex, löst es potenziell eine Immunreaktion aus und kann auf Ähnlichkeiten zu körpereigenen Proteinen untersucht werden. Es wird davon ausgegangen, dass bestimmte Strukturähnlichkeiten zwischen Viren und körpereigenen Stoffen ausreichend sind, damit ein körpereigenes Protein fälschlicherweise an einen MHC-Komplex bindet. Dies wird als »Molecular Mimicry« bezeichnet.
Die Bestimmung der Bindung vieler Proteine an einen MHC-Komplex im Labor ist aufwendig und kostspielig, daher greifen die Wissenschaftler auf Programme der Bioinformatik zurück, um in-silico die Bindungswahrscheinlichkeit zu bestimmen. Hierfür steht eine große Auswahl an Programmen zur Verfügung, die verschiedene Methoden und Algorithmen implementieren.

Im Rahmen dieser Bachelorarbeit wurde eine Software entwickelt, die es den Wissenschaftlern ermöglicht, eine hohe Anzahl an Proteinen systematisch zu untersuchen. Die Software vereint mehrere Programme der Bioinformatik, um die Bindung von Proteinen an MHC-II Moleküle zu bestimmen. Die Ergebnisse der Berechnung der Bindungswahrscheinlichkeit werden dem Benutzer durch Grafiken und Tabellen visualisiert. Durch die Analyse einer großen Anzahl von Proteinen wird durch die Wissenschaftler ein Datenbestand aufgebaut, der durch die Software statistisch ausgewertet werden kann. Die Strukturen und Gemeinsamkeiten, die durch die Software in den Proteinen gefunden werden, können verwendet werden, um Vorhersagen über die Auslösung einer Immunreaktionen durch körpereigene Stoffe zu treffen.

1.2 Zielsetzung

Ziel der Bachelorarbeit ist die Entwicklung einer Software, die es den Wissenschaftlern ermöglicht, eine Vielzahl an Proteinen zu analysieren. Als Basis dienen zwei Programme der Bioinformatik: SYFPEITHI und das »MHC-II binding prediction«-Programm, deren Funktionalität gekoppelt werden soll. Die Ergebnisse der Analysen sollen auf verschiedene Arten visualisiert werden, um die bisher sehr limitierte Ausgabe der vorhandenen Programme zu verbessern. Somit wird eine völlig neue Sicht auf die Daten möglich, die zu neuen Erkenntnissen in der Immunologie-Forschung führen soll.
Die Anwendung soll Proteinsequenzen und auch die Ergebnisse der Bindungsanalysen und statistischen Auswertungen speichern und verwalten. Weiterhin sollen die Ergebnisse der Analysen auf verschiedene Art visualisiert werden. Auf diese Weise sollte den Anwendern ein komfortables Tool zur Verfügung gestellt werden, das die Analyse großer Datenmengen deutlicht erleichtert.
Durch die Analyse größerer Datenmengen werden die Erkennung von Systematiken und idealerweise auch die Bindungseigenschaften für bisher nicht analysierte Proteinsequenzen möglich.

2 Grundlagen

Im folgenden Abschnitt wird auf die biologischen und die Grundlagen der Bioinformatik eingangen. Im Abschnitt Grundlagen der Bioinformatik werden die wissenschaftliche Datenbank NCBI sowie die beiden in die Software integrierten Programme vorgestellt.

2.1 Biologische Grundlagen

Multiple Sklerose (MS) ist eine Autoimmunkrankheit, bei der körpereigene Zellen durch das Immunsystem angegriffen werden (vgl. Felicitas Witte (2011)). Bei einer Erkrankung kommt es zur Entzündung bestimmter Nervenstrukturen, was unter anderem zu Lähmungen und Gefühlsstörungen führen kann. Multiple Sklerose ist chronisch und bisher nicht heilbar, der Krankheitsverlauf erfolgt in Schüben, wodurch die Erkrankung kontinuierlich verstärkt wird.
Die Ursachen, die zu einer Multiple Sklerose Erkrankung führen, sind noch nicht vollständig erforscht. Die Gene haben Einfluss auf die Wahrscheinlichkeit, an MS zu erkranken. Das Krankheitsrisiko erhöht sich, sofern im Verwandschaftskreis bereits jemand an MS erkrankt ist. Aber auch das Vorhandensein bestimmter Gene kann das Risiko erhöhen. So haben Wissenschaftler festgestellt, dass die Gene »HLA-DRB1*1501« und »HLA-DRB5*0101« Einfluss auf die Erkrankung an MS haben.
Verschiedene Umwelteinflüsse, wie z.B. ein Mangel an Vitamin D und das Rauchen, stehen unter Verdacht, das Risiko einer MS-Erkrankung zu erhöhen. Ein Großteil der an Multiple Sklerose Erkrankten trägt das »Eppstein-Barr-Virus« in sich, das als Einflussfaktor für die Erkrankung angesehen wird. Als weiterer möglicher Einflussfaktor gilt eine Herpes-Infektion. Wie genau diese zu eine Erkrankung beitragen, ist noch nicht genau erforscht, man geht jedoch davon aus, dass es nicht nur einen Einflussfaktor gibt, vielmehr verschiedene Faktoren in Wechselwirkung stehen.
Das Immunsystem (vgl. Neumann (2008) und Janeway u. a. (2009)) hat die wesentliche Aufgabe, zwischen körpereigenen und körperfremden Stoffen zu unterscheiden und alles körperfremde zu eliminieren. Es wird zwischen dem angeborenen (unspezifischen) und dem adaptiven Immunsystem unterschieden.
Das angeborene Immunsystem übernimmt die natürliche Schutzfunktion des Körpers. Die Haut bildet z.B. einen äußeren Schutz, der das Eindringen von schädlichen Substanzen verhindert. Ein anderes Beispiel sind die Schleimhäute, die das Körperinnere schützen.

Gelingt es Fremdkörpern trotzdem in den Körper einzudringen, erfolgt eine zweite Abwehr in Form von Entzündungen, Fieber, oder sogenannten Fress- und Killer-Zellen.
Das adaptive Immunsystem richtet sich gegen äußere Krankheitserreger oder Fremdkörper. Teil des adaptiven Immunsystem sind verschiedene Zellen, insbesondere die Granulozyten, Makropagen, dendritische Zellen, natürliche Killerzellen, B- und T-Zellen (vgl. Neumann (2008), Kapitel 1.2). Damit diese Fremdkörper erkennen können, sind an ihrer Oberfläche Rezeptoren vorhanden, die körperfremde Strukturen binden und damit eine Immunreaktion auslösen können. Die für die Abwehr zuständigen B- und T-Lymphozyten (auch als B- und T-Zellen bezeichnet) sind im Blut als weiße Blutkörperchen vorhanden. Die T-Zellen erkennen Fremdstoffe oder virusbefallene Zellen und bilden Abwehrstoffe zu ihrer Bekämpfung.
Bei einer MS-Erkrankung werden fälschlicherweise die Schutzhüllen der Nervenzellen, die Myelinscheiden, als fremd erkannt und durch die T-Zellen angegriffen. Hierbei haften an den Nervenzellen sogenannte »Fresszellen«, die die Nervenhüllen vernichten. Das führt zu Entzündungen des Nervengewebes, das nicht vollständig regeneriert werden kann. Die so entstandenden Schäden führen dazu, dass die Nervenimpulse nicht mehr vollständig weitergeleitet werden und Beeinträchtigungen wie Lähmungen entstehen.
In der Multiple Sklerose Forschung wird untersucht, wann eine T-Zelle eine Immunreaktion auslöst. Damit es zu einer Immunreaktion kommen kann, muss der »Fremdkörper« der T-Zelle präsentiert werden.
Als Haupthistokompatibilitätskomplex (MHC) wird eine Reihe von Genen bezeichnet, die für die Immunerkennung verantwortlich sind. Die MHC-Komplexe des menschlichen Körpers werden auch als »Human Leukocyte Antigen« (HLA bzw. HLA-Typ) bezeichnet. Kodiert werden sie als MHC-Klasse-I- und MHC-Klasse-II-Komplex und befinden sich auf der Oberfläche der Zelle. Über das MHC-Klasse-I-Komplex werden infizierte oder entartete Zellen erkannt, die körperfremde Stoffe erzeugen.
Die MHC-Klasse-II-Komplexe präsentieren den T-Zellen Proteine, worauf diese die Produktion von Antikörpern zum Eleminieren dieser initiieren können.
Proteine sind Ketten von Aminosäuren, die in den Zellen von Lebewesen vorhanden sind. Es gibt verschiedene Darstellungsmöglichkeiten von Proteinen. So kann z.B. ihre Struktur als 3D-Grafik dargestellt werden. Eine weitere Darstellungsmöglichkeit ist die Sequenz von Aminosäuren, wobei das Protein durch eine Zeichenkette repräsentiert wird. Die Aminosäuren werden dabei als Einbuchstabencode (für eine Liste der Codes siehe[1]) kodiert, die Aminosäure Alanin wird durch den Buchstaben »A« dargestellt, die Aminosäure Leucin durch den Buchstaben »L« und als weiteres Beispiel Prolin mit dem Buchstaben »P«. Ein Protein, das aus den Aminosäuren Alanin, Leucin und Prolin besteht, würde als Sequenz folgendermaßen dargestellt werden: »ALP«. Die in Lebewesen vorhandenen Proteine enthalten meist wesentlich mehr Aminosäuren. Ein Protein, das maximal aus 20 Aminosäuren besteht, wird in der Fachliteratur als Peptid bezeichnet.
Damit ein Protein einer T-Zelle präsentiert werden kann und diese ggf. eine Immunreaktion

[1] http://www.mun.ca/biochem/courses/3107/aasymbols.html am 04.07.2011

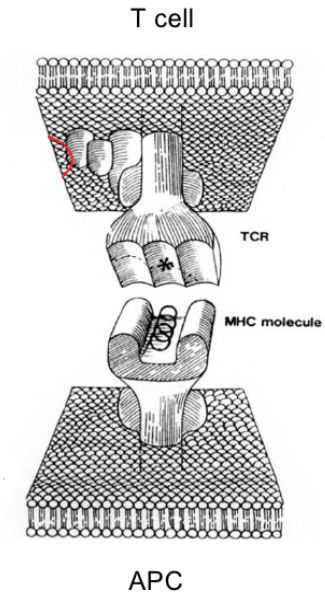


Abbildung 2.1: MHC-II Molekül präsentiert dem T-Zell-Rezeptor ein Peptid

auslöst, muss das Protein zerkleinert werden und an das MHC-II-Molekül binden (vgl Jürgen Groth (2011) Kapitel »Immunologische Erkennung von Selbst und Fremd«). Kommt eine Bindung zustande, wird es an die Oberfläche der Zelle transportiert und der T-Zelle präsentiert.
Abbildung 2.1 zeigt exemplarisch, wie das Protein der T-Zelle durch das MHC-II-Molekül präsentiert wird. Die T-Zelle verfügt über einen T-Zell-Rezeptor (in der Abbildung als TCR, aus dem englischen T-Cell-Receptor, bezeichnet), an den Teile des Proteins ebenfalls binden müssen, damit eine Immunreaktion ausgelöst wird. Das MHC-II-Molekül sitzt in der Abbildung auf der antigenpräsentierenden Zelle (in der Abbildung als APC, aus dem Englischen »Antigene presenting cell«, bezeichnet).
Das zerkleinerte Protein (Peptid) muss an das MHC-II-Molekül sowie an den T-Zell-Rezeptor binden. Abbildung 2.2 zeigt diesen Bindungsvorgang beispielhaft. Das Protein ist als Sequenz in der Einbuchstaben-Kodierung dargestellt. Die Abbildung verdeutlicht, dass nur ein Teil der Aminosäuren für die Bindung an den T-Zell-Rezeptor zuständig sind (in der Abbildung sind das die Aminosäuren F, K, N und T), beziehungsweise an das MHC-II-Molekül binden (in der Abbildung die Aminosäuren F, I und R).
Grund für die Fehlreaktion des Immunsystems kann das »Molecular Mimicry« (Neumann (2008), Abschnitt 6, Seite 212) sein, hierbei sind Ähnlichkeiten zwischen körpereigenen und körperfremden Proteinen vorhanden. Diese können durch gleiche Aminosäuren innerhalb der Proteinsequenz entstehen.

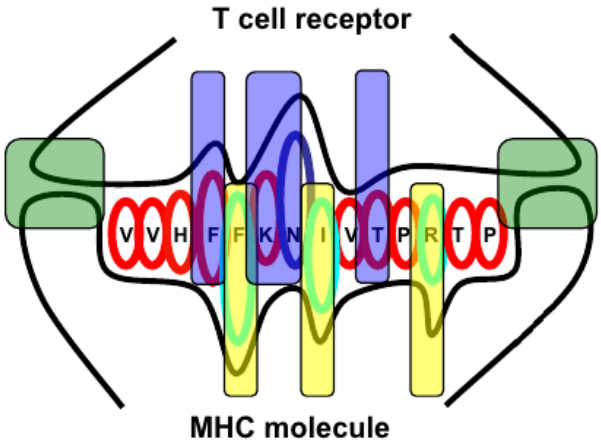


Abbildung 2.2: Bindungsgrube

2.2 Grundlagen der Bioinformatik

Die Bioinformatik ist eine interdisziplinäre Wissenschaft, die Problemstellungen aus der Biologie, Medizin und Chemie mit Hilfe von Methoden der Informatik löst (vgl. Lesk (2003)). Die wesentlichen Bestandteile der Bioinformatik sind die Verwaltung von biologischen Daten sowie die Sequenzanalyse und die Strukturbioinformatik. Ein Beispiel für die Verwaltung von biologischen Daten ist die wissenschaftliche Datenbank »National Center for Biotechnology Information« (NCBI) des National Institute of Health (NIH), die Zugriff auf verschiedene wissenschaftliche Veröffentlichungen und Datenblätter ermöglicht sowie eine Suchfunktion für verschiedene biologische Datenbanken bereitstellt. Bei der Sequenzanalyse werden Protein- oder DNA-Sequenzen auf bestimmte Charakteristika untersucht. Das »MHC-II binding prediction«-Programm und SYFPEITHI basieren auf Sequenzanalysen. In der Strukturbioinformatik wird die Struktur sowie die Interaktion von Molekülen visualisiert und mit Hilfe von Algorithmen berechnet. Die Visualisierung kann in Form von 3D-Grafiken erfolgen, wobei die Strukturen dargestellt werden.
In den folgenden drei Abschnitten wird die Datenbank NCBI vorgestellt sowie das »MHC-II binding prediction«-Programm, das mit Algorithmen der Bioinformatik die Bindungswahrscheinlichkeit eines Proteins an einem MHC-II Molekül berechnet. Im dritten Abschnitt wird SYFPEITHI beschrieben, wobei es sich um eine Software und Datenbank handelt, die verschiedene Anker-Motive bereitstellt und ebenfalls Bindungswahrscheinlichkeiten für Proteinsequenzen und MHC-II Moleküle bestimmen kann.

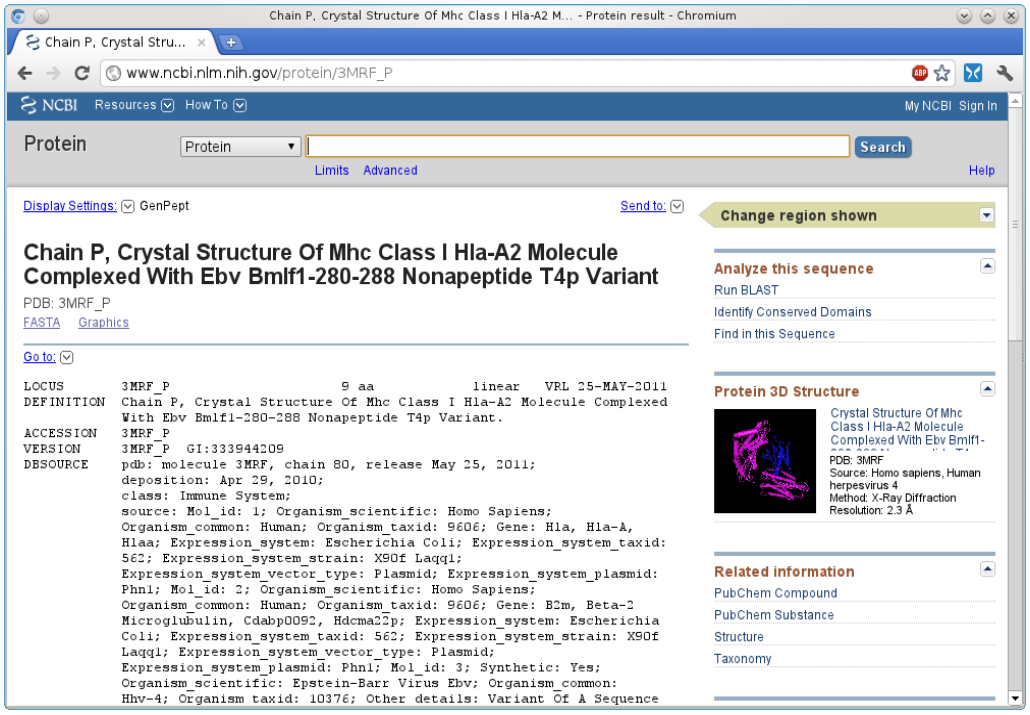


Abbildung 2.3: NCBI Proteindatenbank

2.2.1 Wissenschaftliche Datenbank NCBI

»National Center for Biotechnology Information« (NCBI - siehe ncbi (2011)) ist eine Website, die durch das amerikanische »National Institute of Health«[2] gepflegt wird. Sie ermöglicht den Zugriff auf verschiedene medizinische und biologische Datenbanken und stellt Programme zur Analyse von Proteinsequenzen bereit[3]. Über die Webanwendung PubMed[4] ist die Recherche nach wissenschaftlichen Veröffentlichungen aus Fachzeitschriften möglich, diese können dort eingesehen und heruntergeladen werden.
Die Wissenschaftler des Inims verwenden die Proteindatenbank[5] von NCBI, um Datenblätter von Proteinen zu finden. Aus diesen werden die Proteinsequenzen extrahiert und für in-silico Analysen verwendet.
Abbildung 2.3 zeigt das Datenblatt eines Proteins in der NCBI-Proteindatenbank.

[2] http://www.nih.gov
[3] Eine Übersicht der Dienste, die durch die NCBI bereitgestellt werden, findet sich auf folgender Website http://www.ncbi.nlm.nih.gov/guide/all/
[4] http://www.ncbi.nlm.nih.gov/pubmed/
[5] http://www.ncbi.nlm.nih.gov/protein/

2.2.2 »IEDB binding prediction«-Programm

Das »IEDB binding prediction«-Programm[6] vereinigt verschiedene Methoden der Bioinformatik, um die Bindungswahrscheinlichkeit von Peptiden an MHC-II Moleküle zu berechnen. Die Methode, die nach Aussage der Entwickler der IEDB die besten Resultate für die Berechnung der Bindungswahrscheinlichkeit erzielt, ist der Consensus-Ansatz (vgl. Wang u. a. (2008)).
Der Consensus-Ansatz ist folgendermaßen implementiert:
Für jedes MHC-II Molekül, dessen Bindungswahrscheinlichkeit durch verschiedene Algorithmen bestimmt werden kann, wurden die drei Algorithmen mit der besten Laufzeit ausgewählt. Für die Berechnung der Bindungswahrscheinlichkeit werden diese drei verwendet. Das Ergebnis wird bestimmt, indem der Median der drei Bindungswahrscheinlichkeiten gebildet wird.
Die Algorithmen zur Berechnung der Bindungswahrscheinlichkeit verfolgen verschiedene Ansätze. Sie basieren unter anderem auf statistischen Analysen von Labordaten, bei denen für eine große Menge von Proteinen bereits geprüft wurde, ob sie an das MHC-II-Molekül binden. Um Strukturen und Gemeinsamkeiten in den bindenden Proteinen zu finden, wurde das, aus der künstlichen Intelligenz bekannte, maschinelle Lernen eingesetzt. Dadurch können für bisher nicht bekannte Proteinsequenzen Vorhersagen über die Bindungseigenschaften getätigt werden. Ein weiterer Ansatz ist der Einsatz statistischer Methoden, bei denen Korrelationsmatrizen aufgestellt wurden, um Aussagen zur Bindungswahrscheinlichkeit zutätigen.
Das »IEDB binding prediction«-Programm ist frei herunterladbar von der Website der IEDB[7] und kann als Kommandozeilenprogramm eingesetzt werden. Die Anwendung wird von der IEDB auch als Webinterface[8] bereitgestellt, das jedoch einige Einschränkungen beinhaltet. Die Ausgabe ist auf eine tabellarische Ansicht beschränkt, wobei pro Anfrage die Bindungswahrscheinlichkeit für lediglich ein Protein bestimmt werden kann.

[6] http://tools.immuneepitope.org/analyze/html/mhc_II_binding.html
[7] http://tools.immuneepitope.org/analyze/html/download_MHC_II.html
[8] http://tools.immuneepitope.org/analyze/html/mhc_II_binding.html

Position	**1**	2	3	**4**	5	6	**7**
Aminosäure	I			I			I
	L			F			L
	V			Y			M
							F
							V

Tabelle 2.1: Anker-Motiv des MHC-II Moleküls »HLA_DRB1*1501«

2.2.3 SYFPEITHI

SYFPEIHII[9] (vgl. Rammensee u. a. (1999)) ist eine Webanwendung, über die bestimmte Anker-Motive in einer Proteinsequenz gesucht werden können. Anhand der Anker-Motive kann über die Webanwendung die Bindungswahrscheinlichkeit bestimmt werden. Hierbei werden die Anzahl der passenden Anker-Elemente sowie verschiedene Eigenschaften[10] der Aminosäuren in die Berechnung einbezogen.
Wie im Abschnitt 2.1 erläutert, bindet ein Teil der in der Proteinsequenz enthaltenen Aminosäuren sowohl an das MHC-II-Molekül als auch an den T-Zell-Rezeptor. Die an der Bindung beteiligten Aminosäuren werden als Anker oder auch Bindungsstellen bezeichnet. In der SYFPEITHI-Datenbank sind verschiedene Kombinationen von Aminosäuren hinterlegt, für die im Labor festgestellt wurde, dass sie an das MHC-Molekül binden. Bezeichnet werden diese Einträge als Anker-Motive. Neben den auf Labordaten basierenden Anker-Motiven sind in der Datenbank bereits publizierte Anker-Motive vorhanden.
In der Tabelle 2.1 ist beispielhaft ein Anker-Motiv für das MHC-II Molekül »HLA_DRB1*1501« aufgeführt. Es enthält insgesamt drei mögliche Bindungsstellen, die auch als Anker-Elemente bezeichnet werden. Jedes Anker-Element enthält verschiedene Aminosäuren, die potenziell für eine Bindung an ein MHC-II Molekül in Frage kommen. An den Stellen, an denen keine Aminosäure im Anker-Motiv aufgeführt ist, sind die an die T-Zelle bindenden Aminosäuren vorhanden. Laut dem in der Tabelle 2.1 aufgeführten Anker-Motiv binden an der Position 1 die Aminosäuren Isoleucin (I), Leucin (L) oder Valin (V).

[9] http://www.syfpeithi.de
[10] siehe auch http://www.syfpeithi.de/Scripts/MHCServer.dll/Info.htm#General

3 Verwendete Software-Technologien

In den drei folgenden Abschnitten werden die wichtigsten, für die Realisierung der Software verwendeten, Technologien erläutert. Python und Django implementieren die serverseitige Logik und erzeugen HTML, das an den Browser ausgeliefert wird. jQuery ist eine JavaScript Klassenbibliothek, die eingesetzt wird, um ein möglichst dynamisches Benutzerinterface auf Clientseite zu generieren.

3.1 Programmiersprache Python

Python[1] ist eine Open-Source-Skriptsprache, deren Interpreter auf alle gängigen Betriebsysteme (Windows, Linux, MacOs) portiert ist (weiterführende Literatur zu Django bieten folgende Standardwerke: Lutz (2011) und Alchin (2010b)). Python wurde Anfang der 90er Jahre von Guido van Rossum am »Centrum voor Wiskunde en Informatica« in Amsterdam entwickelt. Die Sprache Python wird permanent weiterentwickelt, Spracherweiterungen und Neuerungen fließen stetig ein.
Der Standard Python-Interpreter, der als CPython bezeichnet wird, ist in der Programmiersprache C implementiert. Jython ist ein Python-Interpreter, der in Java entwickelt wurde und es ermöglicht, Python-Programme in Java-Anwendungen zu integrieren. Python-Skripte können direkt aus Java ausgeführt werden und damit auf Java-Klassen zugreifen. Eine weitere alternative Implementierung stellt IronPython dar. Es handelt sich hierbei um einen Interpreter, der in der Programmiersprache C# für das .Net-Framework von Microsoft entwickelt wurde. Eine interessante Implementierung des Python-Interpreters stellt PyPy[2] dar, der Interpreter ist selbst in Python geschrieben. Mit PyPy können Spracherweiterungen entwickelt werden, die bei der Integration in den CPython-Interpreter aufwändig wären. Laut der Projekt-Homepage sollen Python-Skripte mit dem PyPy Interpreter schneller als mit dem CPython-Interpreter ausgeführt werden können.
Verschiedene Firmen setzen auf Python, so zum Beispiel die Nasa oder Google. Google verwendet in vielen Bereichen Python. Es ist unter anderem möglich, in der Google-»Appengine« Python Programme auszuführen.

[1] http://www.python.org/
[2] http://www.pypy.org

Die Standardbibliothek von Python besitzt einen sehr großen Funktionsumfang. Zusätzlich stehen viele Erweiterungen und Bibliotheken zur freien Verfügung bereit. Populäre Beispiele sind z.B. das Webframework Django, das auch für die Entwicklung dieser Bachelorarbeit eingesetzt wird, oder das »Networking Framework«-Twisted, das verschiedene Netzwerkprotokolle implementiert und sie über eine API bereitstellt.
Für die Entwicklung von Desktop-Anwendungen stehen Bibliotheken zur Verfügung mit denen grafische Oberflächen programmiert werden können. Es existieren zum Beispiel Python-Bibliotheken für die GUI-Frameworks Qt und Gtk+.
Der Quellcode von Python-Skripten wird vor der Ausführung durch den CPython-Interpreter in Bytecode übersetzt, was sich positiv auf die Laufzeit auswirkt.
Python ermöglicht eine objektorientierte Programmierung und basiert auf dynamischer Typisierung. Ein markantes Merkmal gegenüber anderen gängigen Programmiersprachen ist, dass die Syntax auf Einrückungen basiert. Schleifenrümpfe, oder Inhalte von konditionalen Anweisungen werden durch Einrückung gruppiert. Im Gegensatz zu Programmiersprachen wie C, C++ oder Java, die hierfür geschweifte Klammern verwenden.
Python Programme werden in Module (dargestellt durch Dateien im Dateisystem) aufgeteilt und durch »imports« eingebunden. Python unterstützt »Namespaces«, wodurch eine klare Trennung zwischen den Modulen stattfindet und Kollisionen im Namensraum vermieden werden.

3.2 Django Webframework

Django ist ein in Python entwickeltes Webframework, das auf dem »Model view controller«-Prinzip (MVC, siehe Reenskaug (1979)) basiert. (Weiterführende Literatur zu Django bieten folgende Standardwerke: Holovaty (2010), Bennett (2008), Alchin (2010a))
Initial wurde Django für die News-Website »Lawrence Journal-World«[3] entwickelt und im Jahre 2005 unter der BSD-Lizenz Open-Source gestellt. Seitdem wird Django von einem wachsenden Entwicklerkreis weiterentwickelt.
Die MVC-Terminologie wurde durch die Django-Entwickler anders als ursprünglich gedacht interpretiert (siehe Foundation (2011)). In Django erzeugt die View HTML, das an den Benutzer ausgeliefert wird. Im Gegensatz zum MVC Grundgedanken beschreibt sie nicht, wie die Daten präsentiert werden, sondern dient zur Erzeugung der Datenrepräsentation. Django erzeugt die Datenpräsentation mit Hilfe von Templates. Der Controller ist das Framework selbst, das anhand der URL-Konfiguration die jeweilige View aufruft. Die Model-Klassen und die Datenbankschicht sind über ein objektrelationales Mapping implementiert.
Django vereinfacht typische Aufgaben der Webentwicklung. Die Erzeugung des HTML-Codes wird durch eine Template-Engine vereinfacht. Eine häufige Aufgabe ist das Program-

[3] http://www.lawrence.com

mieren und Validieren von Formularen. Mit Django können Klassen definiert werden, anhand derer Formulare erzeugt werden, die wiederum die Validierung der Eingaben übernehmen. Eine weitere Möglichkeit ist die Erzeugung der Formulare anhand der Klassen für das objektrelationale Mapping, hierbei werden die Attribute der Datenbanktabelle als Formular dargestellt. Die Formulare können verwendet werden, um neue Daten in die Datenbank einzupflegen oder bestehende Daten zu verändern.

Entwicklung einer Django-Anwendung

Die Grundstruktur einer Django-Anwendung ist in Abbildung 3.1 gezeigt. Sie besteht im Wesentlichen aus der URL- und der Anwendungs-Konfiguration. Die URL-Konfiguration wird in der Datei »urls.py« festgelegt und enthält die Zuordnung der URLs zu verschiedenen »Views«. Diese verarbeiten den Http-Request und erzeugen über ein Template dynamisch HTML-Code. Die URL-Konfiguration wird durch den »Request-Handler« von Django gelesen, der dann entsprechend die Views der Anwendung aufruft.
In der Anwendungs-Konfiguration, die in der Datei »settings.py« enthalten ist, werden verschiedene Parameter für die Webanwendung definiert. Unter anderem werden dort die Zugangsdaten für die Datenbank oder die installierten Apps festgelegt. Neben diesen Konfigurationsdateien enthält eine Django-Webanwendung noch mehrere »Apps«, die die Anwendungslogik implementieren. Eine »App« ist eine möglichst modulare Komponente der Webanwendung.
Die Grundstuktur eines Django-Projekts wird mit dem Kommandozeilenprogramm »django-admin.py« durch folgenden Aufruf erzeugt: »django-admin.py startproject <name>«. In dem damit erstellten Projektverzeichnis ist das Kommandozeilenprogramm »manage.py«, mit dem die einzelnen »Apps« angelegt werden können, enthalten.
Eine »App« wird mit dem Befehl »manage.py startapp <appname>« im Projektverzeichnis erzeugt. Diese besteht aus den einzelnen Views, die in der Datei »views.py« definiert werden, und aus mehren HTML-Templates. Die Klassen für den Datenbankzugriff sind in der Datei »models.py« als fester Bestandteil der App hinterlegt.
Abbildung 3.2 zeigt zwei Beispiele für die Struktur einer »App«. Die Pfeile in den Grafiken verdeutlichen den Zugriff der einzelnen Module. Die View greift auf das Template und die Datenbank-Klassen zu. Die als »App2« bezeichnete App enthält zusätzlich eine App-spezifische URL-Konfiguration, in der die Weiterleitung der Anfragen an die jeweilige View festgelegt wird.
Die Entwicklung einer Django-App soll am Beispiel einer Komponente gezeigt werden, die Proteinsequenzen aus der Datenbank laden und anzeigen kann.
Die Datenbanktabelle, in der die Proteinsequenzen gespeichert werden, wird als Klasse in der Datei »models.py« angelegt. Es handelt sich um eine Klasse für das objekrelationale Mapping, über die Datensätze geladen, gelöscht und verändert werden können. Listing 3.1

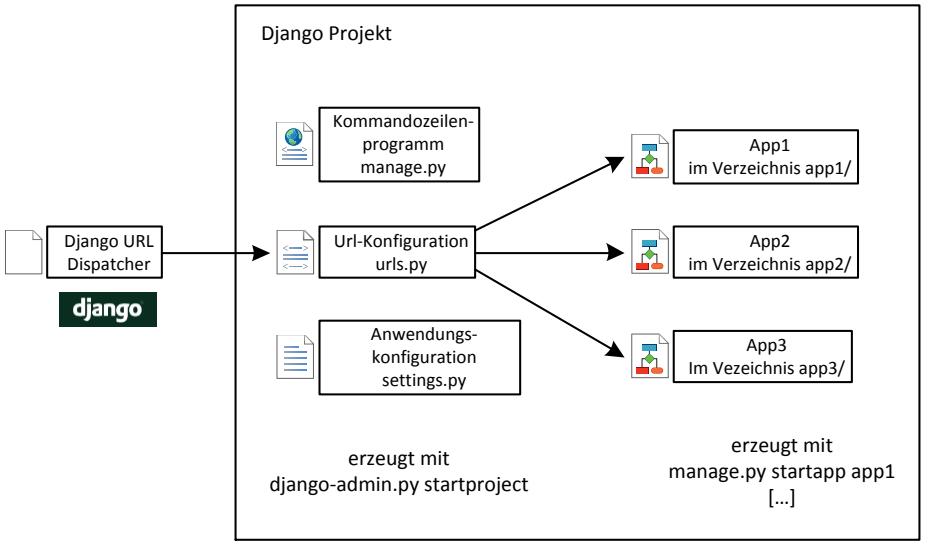


Abbildung 3.1: Grundstruktur eines Django-Projektes

zeigt eine vereinfachte Model-Klasse, die die Tabelle »Sequence« darstellt. Das Attribut »seq« speichert die Proteinsequenz als Zeichenkette.

```
1 class Sequence(Model):
2     seq = TextField()
```

Listing 3.1: Auszug aus der Datei »models.py«

Eine Instanz der Klasse repräsentiert einen Datensatz in der Datenbank. Zum Erzeugen eines Datensatzes wird die Klasse instanziiert, das Attribut »seq« gesetzt und über die Methode »save()« der Oberklasse »Model« in der Datenbank gespeichert. Zum Laden aller Datensätze der Tabelle »Sequence«, würde folgender Aufruf verwendet werden: »Sequence.objects.all()«. Im Abschnitt 6.8 wird der Datenbankzugriff mit der API für das objektrelationale Mapping ausführlich erläutert.

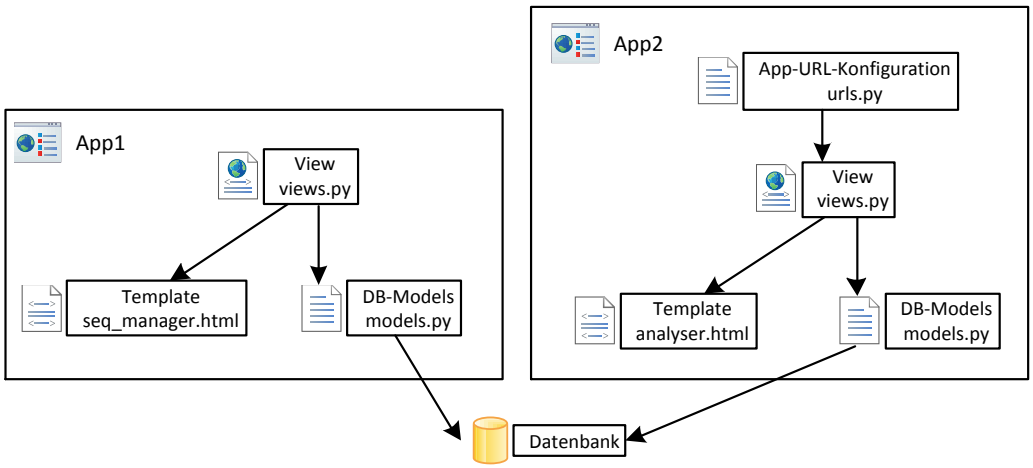


Abbildung 3.2: Struktur einer »Django App«

```
1 def seq_manager(request):
2     sequences = Sequence.objects.all()
3     return render_to_response('seq_manager/seq_manager.html',
4                               { 'sequences': sequences })
```

Listing 3.2: Auszug aus der Datei »views.py

Die »Views« zum Erzeugen von HTML-Code werden in der Datei »views.py« angelegt, Listing 3.2 enthält die View »seq_manager()«.
Jede »View« nimmt den Parameter »request« entgegen, wobei es sich um eine Instanz der Klasse »HttpRequest« handelt. Über diese kann auf den Http-Request und die per Http-GET oder Http-POST übergebenen Parameter zugegriffen werden.
Die View »seq_manager« lädt über das Model »Sequence« alle Datensätze der Tabelle »Sequence« aus der Datenbank (Listing 3.2, Zeile 2). Die Methode »render_to_response()« erzeugt anhand eines Templates HTML-Code, der an den Aufrufer zurückgegeben wird. Als zweiten Parameter nimmt »render_to_response()« eine Datenstruktur entgegen, die die Namen und Inhalte der Parameter enthält, auf die im Template zugegriffen werden kann.
Listing 3.3 zeigt das Template, das in der View verwendet wird. In Zeile 4 wird ein Template-Tag zum Iterieren über alle Elemente der Variable »sequences« verwendet. In jeder Iteration wird die Variable »sequence« mit dem aktuellen Objekt der Liste gesetzt. In der Zeile 5 wird der Inhalt der Klassenvariablen »seq« des »Sequence«-Objekts umschlossen von

HTML-Tags ausgegeben.

```
1  <html>
2  <head><title>Sequence Manager</title></head>
3  <body>
4  {% for sequence in sequencess %}
5  <p>{{ sequence.seq }}</p>
6  {% endfor %}
7  </body>
8  </html>
9  </body>
10 </html>
```

Listing 3.3: Template des Sequence Managers

Djangos Template API stellt wesentlich mehr Tags zur Verfügung als die in Listing 3.3 verwendeten. Innerhalb eines Templates können weitere Templates eingebunden werden oder Templates können von Templates »erben«. Bei der Template-Vererbung wird ein »Master«-Template angelegt, das verschiedene Blöcke enthält, die von dem erbenden Templates überschrieben und mit Inhalt gefüllt werden.
Um das Beispiel zu komplettieren, muss es dem Benutzer ermöglicht werden, die View »seq_manager()« aufzurufen. Die Zuordnung einer URL zu einer View erfolgt über die URL-Konfiguration »urls.py« die in Listing 3.4 dargestellt ist.

```
1 urlpatterns = patterns('',
2     #[...]
3     (r'^/seq_manager/$', 'src.sequence_manager.views.seq_manager'),
4     # [...]
5 )
```

Listing 3.4: Auszug aus der Datei »urls.py«

Die Variable »urlpatterns« wird von Django verwendet, um die Zuordnung der URLs auf die Views vorzunehmen. Die Initialisierung der Variablen erfolgt über den Aufruf der Methode »patterns()«, die eine variable Anzahl an Parametern entgegennimmt und eine Liste mit »RegexURLResolver«-Objekten zurückgibt. Diese werden beim Verarbeiten des Http-Requests zum Aufrufen der entsprechenden »View« verwendet.
Der erste Parameter ist ein optionaler Präfix, der dem Modulpfad zur View vorangestellt wird. In dem Beispiel ist er leer. Würden mehrere URLs in der URL-Konfiguration eingetragen werden, deren Views in dem Modul »src.sequence_manager« vorhanden sind, könnte der Präfix auf »src.sequence_manager« gesetzt werden. Dies hätte zur Folge, dass nur noch der Name der View und nicht mehr der gesamte Modulpfad angegeben werden müsste.
Als weiteren Parameter nimmt die Methode »patterns()« eine beliebige Anzahl an Zweier-Tupeln entgegen. Jedes Tupel enthält als erstes Element einen regulären Ausdruck, der die

URL beschreibt. Das zweite Element des Tupels ist die View, die den Aufruf entgegennimmt und verarbeitet.

Sessions und Benutzerauthentifizierung

Da Http ein zustandsloses Protokoll ist, wurde es bereits 1994 um Http-Cookies erweitert. Django übernimmt die Verwaltung von Cookies, es setzt und überprüft die Cookies und bietet eine API, um benutzerspezifische Informationen abzulegen. Die gespeicherten und einem Benutzer zugeordneteten Daten werden in der Django-Terminologie als »Session« bezeichnet. Ein typischer Anwendungsfall wäre der Warenkorb eines Webshops, bei dem jeder Benutzer seinen eigenen Warenkorb mit Produkten füllen kann.
Bei der Verwendung von Sessions sind die Benutzer zwar identifiziert, jedoch noch anonym. Für die Authentifizierung von Benutzern mit Benutzername und Passwort wird eine API von Django bereitgestellt, die diese in der Datenbank anlegt. Den Benutzern können Berechtigungen zugewiesen werden, die ebenfalls in der Datenbank hinterlegt werden. In den Views kann geprüft werden, ob ein Benutzer berechtigt ist, diese aufzurufen.

Sicherheit

Ein weiterer Vorteil von Django ist, dass mehrere Sicherheitsmechanismen implementiert sind. Ein häufiges Problem in Webanwendungen ist die SQL-Injection, wobei Formularangaben ungefiltert an die Datenbank in Form eines SQL-Statements weitergegeben werden und so ein Angreifer sich durch Manipulation den Zugriff auf die Datenbank verschaffen kann. Djangos Datenbank-API schützt vor solchen Attacken, indem die Eingaben gefiltert werden. Ein weiterer typischer Angriff auf Webanwendungen ist Cross-Site-Scripting, wobei ein Angreifer JavaScript-Code in die Website einbettet. Dieser wird beim nächsten Aufruf der Website ausgeführt und kann zum Stehlen von Cookies oder zur Datenmanipulation mit den Berechtigungen des jeweiligen Aufrufers verwendet werden[4]. Falls Formulareingaben ungefiltert in der Datenbank gespeichert und im HTML wieder ausgegeben werden, kann ein Angreifer seinen Code ebenfalls einschmuggeln. Dies erfolgt zum Beispiel über ein Formular zum Senden eines Kommentars für ein Blog-Post. Der Angreifer gibt anstelle eines ernsthaften Kommentars JavaScript-Code ein und lässt es in der Datenbank speichern. Beim nächsten Aufruf des Kommentars wird der Code des Angreifers an den Browser des Aufrufers ausgeliefert und ausgeführt. Die Django Template-Engine schützt vor solchen Attacken, indem bei der Ausgabe von Variablen im Template der Inhalt gefiltert wird. Dabei erfolgt die

[4] Dieser Angriff wird als »Cross Site Request Forgery« bezeichnet. Django stellt eine Erweiterung zum Schutz davor bereit, die in Version 1.3 fest integriert ist

Ausgabe so, dass der JavaScript-Code angezeigt, anstatt in die Website eingebettet und ausgeführt wird.

Integration in einen Webserver

Für den Betrieb einer Django-Webanwendung wird ein Webserver benötigt, der es ermöglicht, Python-Code auszuführen. In den Anfängen des Webs erfolgte das über die »Common Gateway Interface«-Schnittstelle (CGI). Hierbei ruft der Client ein Programm auf, das auf dem Server ausgeführt wird. Die Anfrage wird durch das Programm verarbeitet und die Ausgabe des Programms an den Aufrufer zurückgegeben. Entscheidender Nachteil dabei ist, dass für jede Anfrage ein Prozess gestartet wird. Der Aufruf wird dadurch wesentlich verlangsamt.
Die heute gängigen Webserver verfügen über Module, die mehrere Prozesse starten, um parallel Anfragen zu verarbeiten. Der weit verbreitete Webserver Apache[5] stellt Module für die Programmierspachen Python (mod_python) und PHP (mod_php) bereit. Hierbei erzeugt der Apache-Webserver einen Prozess mit dem jeweiligen Sprachinterpreter. Die Python- oder PHP-Programme werden als Unterprozess des Webservers ausgeführt, was einen Geschwindigkeitsvorteil zur Folge hat. Nachteilig ist, dass die Skripte unter den gleichen Berechtigungen wie der Webserver-Prozess laufen, wodurch Sicherheitsprobleme entstehen können. Um eine möglichst hohe Kompatibilität zwischen Python Programmen, Python-Frameworks und verschiedenen Webservern zu erlangen, wurde das »Python-Web-Server-Gateway-Interface« (WSGI) entwickelt. Es ist als Python »Python Enhancement Proposal« (siehe Eby (2011)) spezifiziert und definiert das Protokoll, über das der Webserver mit der Python-Anwendung kommuniziert. In Django ist dafür der WSGI-Handler integriert, somit kann eine Django-Anwendung mit jedem WSGI-kompatiblen Webserver betrieben werden.
Abbildung 3.3 stellt die Integration einer Django-Anwendung mit einem Webserver dar. Der Webserver nimmt einen durch den Browser des Benutzers initiierten Http-Request entgegen und leitet diesen an die Django-Anwendung weiter. Die Kommunikation zwischen der Django-Anwendung und dem Webserver erfolgt über das WSGI-Protokoll. Djangos WSGI-Requesthandler, nimmt die Anfrage entgegen und leitet sie an das Modul zur Verarbeitung des Http-Request weiter. Das Modul verabeitet die Anfrage und erzeugt eine Antwort, die der Webserver als Http-Response an den Browser zurückgibt.

[5] http://httpd.apache.org

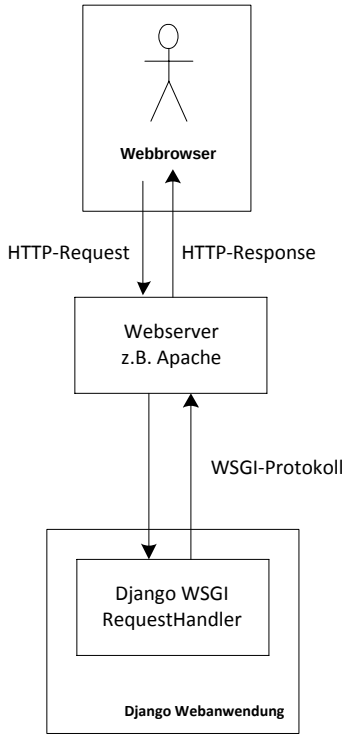


Abbildung 3.3: Integration einer Django-Anwendung in einen Webserver

3.3 jQuery JavaScript-Klassenbibliothek

jQuery[6] ist eine JavaScript Klassenbibliothek, die Klassen und Methoden bereitstellt, um die JavaScript-Entwicklung zu vereinfachen (vgl. Bibeault und Katz (2010)). Anfang 2006 wurde das erste Release von jQuery durch den Entwickler John Resing unter einer Open-Source-Lizenz veröffentlicht. Seitdem hat sich der Entwicklerkreis vergrößert und jQuery wird stetig weiterentwickelt.
In den letzten Jahren haben sich verschiedene JavaScript-Bibliotheken etabliert. Neben jQuery sind auch die Bibliotheken Prototype[7], Dojo Toolkit[8] und Yahoo! UI Library (YUI)[9] weit verbreitet.
Der Anspruch der Benutzer an Webanwendungen ist über die Jahre rapide gestiegen. Mitt-

[6] http://www.jquery.com
[7] http://prototypejs.org/
[8] http://dojotoolkit.org/
[9] http://developer.yahoo.com/yui/

lerweile sind die meisten Webanwendungen sogenannte »Rich Internet Applications«, die von der Bedienbarkeit Desktop-Anwendungen ähneln. Solche »Rich Internet Applications« ermöglichen zum Beispiel »drag and drop« oder simulieren vom Desktop bekannte Elemente wie Fenster, die innerhalb der Anwendung verschoben werden können. Weitere typische Implementierungen sind das animierte Ein- und Ausblenden von Elementen oder das Nachladen von Inhalten vom Server, ohne dass die komplette Website neu geladen werden muss. Für die Bewerkstelligung dieser Aufgaben bieten die JavaScript-Bibliotheken wiederverwendbare Methoden, die diese Funktionalität implementieren.
Die JavaScript Methoden werden direkt mit dem HTML-Code vermischt. Um zum Beispiel in einem » «-Tag eine Methode zu hinterlegen, die beim Anklicken aufgerufen wird, würde folgender Code verwendet werden: » «. Hierbei findet eine starke Vermischung von Code, der für die Anzeige zuständig ist, und Code, der für die Anwendungslogik vorhanden ist, statt. jQuery umgeht dieses Problem.
Durch die Verwendung von jQuery werden JavaScript und HTML nicht mehr vermischt. jQuery unterstützt verschiedene Selektoren, anhand derer auf HTML-Elemente des »Document Object Model«-Baums (DOM) zugegriffen werden kann. Die Elemente können anhand der Inhalte des »id«- oder »class«-Attributs selektiert werden. Alternativ ist es unter anderem möglich, sie anhand ihres aktuellen Zustands, zum Beispiel alle aktiven »Checkboxes« oder anhand ihrer Position, zum Beispiel jede zweite Zeile innerhalb einer Tabelle, zu selektieren.

Programmieren mit jQuery

Für die selektierten HTML-Elemente können Callbacks registriert werden, die bei einem vordefinierten Ereignis aufgerufen werden. Listing 3.5 zeigt einen Code-Ausschnitt, der für ein Element mit dem »id«-Attribut »show_alert« den Callback »show_message()« registriert. Dieser wird beim Klicken auf das Element aufgerufen. Das Element wird, wie es bei CSS ebenfalls der Fall ist, über den Inhalt des »id«-Attributs mit einer Raute (#) vorangestellt referenziert. Die Methode, die beim Click aufgerufen werden soll, wird als Parameter an die Funktion ».click()« übergeben. Hierbei handelt es sich um eine Methode der jQuery-API.

```
1 function show_message() {
2     alert('Das Element wurde geklickt');
3 }
4
5 $('#show_alert').click(show_message);
```

Listing 3.5: Registrierung eines Callbacks

Neben der Registrierung verschiedener Callbacks bietet jQuery noch wesentlich mehr Funktionalität. In der API sind Methoden vorhanden um Http-Requests zu erzeugen und Inhalte

dynamisch nachzuladen, was als »Asynchronous JavaScript and XML« (Ajax) bezeichnet wird. Mit Ajax können Inhalte vom Server nachgeladen werden, wenn die Seite bereits dargestellt ist. Problematisch bei der Erzeugung von Http-Requests über JavaScript ist, dass die verschiedenen Browser keine einheitliche JavaScript-API bereitstellen, es muss daher geprüft werden, welcher Browser verwendet wird und entsprechend browserspezififischer Code ausgeführt werden. Dieses Problem löst jQuery und bietet eine einheitliche Programmierschnittstelle an.
Der Funktionsumfang von jQuery umfasst sogenannte »Utility«-Methoden. Dabei handelt es sich unter anderem um Methoden, die XML-Dateien verarbeiten können, um auf Elemente zuzugreifen oder den gesamten XML-Baum zu traversieren. Neben der Verarbeitung von XML sind Methoden zum Auslesen des Browsernamens, der Bestimmung Bildschirmauflösung des Benutzers und weiteres vorhanden.
Die API von jQuery ermöglicht es, verschiedene Effekte zu verwenden, um Elemente animiert einzublenden, auszublenden, zu verschieben oder zu vergrößern bzw. zu verkleinern.

jQuery Plugins

Ein weiteres Merkmal, das zur hohen Verbreitung von jQuery beiträgt, ist die Erweiterbarkeit durch Plugins. Die durch Plugins implementierten Methoden können verwendet werden, als wären sie Bestandteil der jQuery API.
Für jQuery sind viele verschiedene Plugins Open-Source verfügbar. Bei jQueryUI handelt es sich um ein Plugin, das Elemente, die aus Desktop-Anwendungen bekannt sind, für die Webentwicklung bereitstellt. Angeboten werden eine Reihe von verschiedenen Darstellungselementen, sogenannte »Widgets«. Bei den »Widgets« handelt es sich unter anderem um aufklappbare Kalender, Fortschrittsbalken oder Dialogfenster, die innerhalb des Browserfensters angezeigt werden und durch die API animiert werden können.

4 Analyse

In den folgenden Abschnitten werden die Anforderungen an die zu entwickelnde Software in Form einer Ist-Analyse, Soll-Konzept und einer Anforderungsanalyse verdeutlicht. Der letzte Abschnitt enthält mehrere Vorüberlegungen, die vor Beginn der Entwicklung getroffen wurden.

4.1 Ist-Analyse

Die für die in-silico Experimente benötigten Proteinsequenzen beziehen die Wissenschaftler aus der über die NCBI-Website zur Verfügung gestellten Proteindatenbank. Zum Durchführen einer Analyse müssen die Wissenschaftler das Datenblatt aus der Proteindatenbank laden und die darin aufgeführte Proteinsequenz extrahieren.
Anhand der Proteinsequenz wird in-silico die Bindungswahrscheinlichkeit an ein MHC-Molekül bestimmt. Hierfür verwenden die Wissenschaftler das in Abschnitt 2.2.2 beschriebene »MHC-II binding prediction«-Programm, das von der IEDB[1] als Webanwendung bereitgestellt wird. Das Laden des Datenblatts, das Extrahieren des Proteins und die Analyse mit dem »MHC-II binding prediction«-Programm müssen für jedes zu untersuchende Protein durchgeführt werden.
Als Ergebnis liefert das »MHC-II binding prediction«-Programm eine Menge von Peptidsequenzen und deren Bindungswahrscheinlichkeiten an das MHC-II-Molekül. In einer Proteinsequenz mit insgesamt 415 Aminosäuren sind etwa 400 Peptid-Variationen enthalten. Relevant sind für die Wissenschaftler die Peptide für die eine hohe Bindungswahrscheinlichkeit bestimmt wurde. Diese müssen aus den 400 Peptid-Variationen herausgefiltert werden. In einer Excel-Datei werden die Peptide mit der jeweiligen berechneten Bindungswahrscheinlichkeit gespeichert.
Anschließend werden die Peptide mit der Webanwendung SYFPEITHI auf das Vorhandensein bestimmter Kombinationen von Aminosäuren analysiert. Die Ergebnisse dieser Motivsuche werden ebenfalls mit Excel dokumentiert.
Das »Eppstein-Barr-Virus« hat insgesamt 80 verschiedene Proteine, die durchschnittlich ca.

[1] http://www.immuneepitope.org/

400 Aminosäuren enthalten. Die Analyse dieses Virus würde einen hohen Verwaltungsaufwand nach sich ziehen: Für jedes Protein existieren ca. 400 Peptid-Variationen, d.h. für das gesamte Virus insgesamt 32 000. Aus den 32 000 Peptiden müssen alle Peptide herausgefiltert werden, die eine geringe Bindungswahrscheinlichkeit haben. Diese werden wiederum einzeln mit dem SYFPEITHI Programm analysiert.
Das Beispiel zeigt, dass eine in-silico Analyse des »Eppstein-Barr-Virus« mit einen extrem hohen Verwaltungsaufwand verbunden und praktisch kaum durchführbar ist.

4.2 Soll-Konzept

Die im Rahmen dieser Arbeit zu entwickelnde Software soll umfangreiche Analysen von Proteinsequenzen (wie zum Beispiel dem »Eppstein-Barr-Virus«) ermöglichen und die in der Ist-Analyse beschrieben Arbeitsschritte vereinfachen. Die durch die Analysen erzeugten Daten werden in einer Datenbank gespeichert, womit die Verwaltung der Analyseergebnisse mit MS-Excel hinfällig wird.
Die Anwendung soll Datenblätter direkt aus der NCBI-Proteindatenbank laden und die Proteinsequenz extrahieren können. Das geladene Datenblatt sowie die extrahierte Proteinsequenz werden in einer Datenbank gespeichert. Für einen späteren Zugriff werden den importierten Daten Kategorien und Unterkategorien zugewiesen.
Die importierten Proteinsequenzen werden mit dem »MHC-II binding prediction«-Programm analysiert und die Ergebnisse dem Benutzer tabellarisch und grafisch präsentiert. Hierzu wird die Ausgabe des »MHC-II binding prediction«-Programms verbessert und dem Benutzer verschiedene Visualisierungsmöglichkeiten angeboten.
Eine Funktion zum Erkennen der durch SYFPEITHI vorgegebenen Anker-Motive muss in die Software integriert werden. Die Peptide, für die mit dem »MHC-II binding prediction«-Programm eine gute Bindungswahrscheinlichkeit bestimmt wurde, werden in der Datenbank gespeichert und auf die Existenz der Anker-Motive analysiert. Die Ergebnisse der Motiv-Suche werden dem Benutzer präsentiert und gespeichert.
Der erzeugte Datenbestand gut bindender Peptide, die die gesuchten Anker-Motiven enthalten, soll statistisch auf Aminosäuremuster und Bindungssystematiken analysiert werden.
Die Anwendung muss eine Exportfunktion für das MS-Excel-Format bereitstellen, damit die Analyseergebnisse exportiert und für wissenschaftliche Publikationen aufbereitet und verwendet werden können.
Damit die Rechenleistung eines zentralen Servers mehreren Benutzern zur Verfügung gestellt werden kann, sollte die Software als Webanwendung realisiert werden. Der Vorteil ist, dass für den Zugriff lediglich ein Webbrowser benötigt wird, der für alle gängigen Betriebsysteme verfügbar ist. Ein standortunabhängiger Zugriff auf die Daten und das Durchführen von Analysen wird ebenfalls möglich. Die Wissenschaftler sind nicht an einen bestimmten Computer gebunden, auf dem die Software lokal installiert ist.

4.3 Anforderungsanalyse

Aus dem im Abschnitt 4.2 beschriebenen Soll-Konzept ergeben sich verschiedene funktionale und nicht-funktionale Anforderungen:

Funktionale Anforderungen

A1 **Importieren von Protein-Datensätzen aus der NCBI-Proteindatenbank**
Unter Angabe einer Url der NCBI-Proteindatenbank wird das Datenblatt des referenzierten Proteins heruntergeladen und in die Datenbank der Anwendung importiert. Hierbei wird die Proteinsequenz aus dem Datenblatt extrahiert.

A2 **Einteilen der Protein-Datensätze in Kategorien und Unterkategorien**
Über die Benutzeroberfläche kann jedem importierten Protein-Datensatz eine Kategorie und eine Unterkategorie zugeordnet werden.

A3 **Analyse eines Proteins mit dem »MHC-II binding prediction«-Programm**
Die aus der Proteindatenbank importierte oder durch den Benutzer direkt eingegebene Proteinsequenz kann mit dem »MHC-II binding prediction«-Programm analysiert werden. Dabei werden die Analyseergebnisse auf verschiedene Arten visualisiert.

A4 **Verwalten der importierten Protein-Datensätze**
Die aus der NCBI-Proteindatenbank importierten Protein-Datensätze werden über eine spezielle Bedienoberfläche verwaltet. Die in der Datenbank gespeicherten Protein-Datensätze werden anhand ihrer Kategorien und Unterkategorien angezeigt. Datensätze können gelöscht oder für Analysen selektiert werden.

A5 **Exportieren der gespeicherten Protein-Datensätze**
Über eine Verwaltungsoberfläche werden die in der Datenbank gespeicherten Protein-Datensätze in das MS-Excel-Format exportiert.

A6 **Analyse mehrerer Proteinsequenzen mit dem »MHC-II binding prediction«-Programm**
Über die Verwaltungsoberfläche können mehrere Proteinsequenzen ausgewählt und mit dem »MHC-II binding prediction«-Programm gleichzeitig analysiert werden.

A7 **Visualisierung der Analyseergebnisse**
Die Ergebnisse der Analyse mit dem »MHC-II binding prediction«-Programm werden in tabellarischer und grafischer Form visualisiert. Die tabellarische Form zeigt die Bindungswahrscheinlichkeiten der einzelnen Peptide der Proteinsequenz. Die grafische Visualisierung zeigt verschiedene Grafiken zur Veranschaulichung der Bindungswahrscheinlichkeiten.

A8 **Exportieren der Analyseergebnisse**
Die Ergebnisse der Analyse mit dem »MHC-II binding prediction«-Programm und dem Motif-Finder sollen in das Microsoft-Excel-Format übertragen werden.

A9 **Verwalten der SYFPEITHI-Anker-Motive**
In der Benutzeroberfläche müssen SYFPEITHI-Anker-Motive angelegt und gelöscht werden können.

A10 **Erkennen der SYFPEITHI-Anker-Motive**
Eine Komponente zum Finden der SYFPEITHI-Anker-Motive wird integriert. Die gut bindenden Peptide werden auf das Vorhandensein der SYFPEITHI-Anker-Motive untersucht.

A11 **Statistische Analyse**
Die Peptide, die gut an das MHC-Molekül binden und auf die die SYFPEITHI-Anker-Motive passen, werden statistisch untersucht. Hierbei wird die Verteilung der Aminosäuren auf verschiedene Arten analysiert:

- Die Häufigkeitsverteilung der in den Peptiden vorhandenen Aminosäuren.
- Die Häufigkeitsverteilung der Aminosäuren zwischen den Anker-Elementen.
- Die Häufigkeitsverteilung der in den Anker-Elementen vorhandenen Aminosäuren.
- Die Häufigkeitsverteilung der Aminosäuren an den einzelnen Positionen der Peptide.

Folgende Anforderungen werden an die Implementierung der Anwendungslogik gestellt:

AI1 **Import von Proteinsequenzen aus der NCBI-Proteindatenbank**
Damit die Software hinter einer Firewall oder einem Http-Proxy verwendet werden kann, muss der Import der Proteinsequenzen über das Http-Protokoll erfolgen.

AI2 **Speichern der Daten in einem Datenbanksystem**
Die importierten und durch die Analysen erzeugten Daten sollen in einem Datenbanksystem gespeichert werden. Somit wird Datenkonsistenz und Performanz beim Zugriff auf die Daten gewährleistet.

Nicht-Funktionale Anforderungen

NF1 **Entwicklung als Webanwendung**
Damit die Wissenschaftler standortunabhängig und ohne Installationsaufwand mit der Software arbeiten können, muss diese als Webanwendung realisiert werden.

NF2 **Austausch des Analyseprogramms**
Das Analyseprogramm muss innerhalb der Software so gekapselt werden, dass es bei

Bedarf durch ein anderes ersetzt oder aktualisiert werden kann. Zwischen der Anwendung und dem Analyseprogramm muss eine lose Kopplung vorhanden sein.

NF3 **Plattformunabhängiger Zugriff**
Der Zugriff auf die Anwendung sollte möglichst plattformunabhängig sein. Die Wissenschaftler arbeiten mit verschiedenen Systemen, sei es Windows, Apple Mac OS oder Linux. Die Software darf nicht an einen bestimmten Browser gebunden sein, möglichst viele verschiedene Browser müssen mit der Software kompatibel sein.

NF4 **Erweiterbarkeit**
Die Erweiterbarkeit der Software soll gewährleistet sein, Änderungswünsche oder neue Anforderungen müssen möglichst einfach in die Software integriert werden können.

Nicht gefordert

N1 **Benutzer und Berechtigungssystem**
Der Benutzerkreis ist bisher auf die Wissenschaftler des ZMNH beschränkt, daher ist ein Benutzerverwaltung / Benutzersystem und ein Rechtesystem nicht gefordert. Die Analysen sind für jeden Benutzer einsehbar, es findet keine Trennung zwischen den unterschiedlichen Benutzern statt.
Sobald die Software einem erweiterten Benutzerkreis zur Verfügung gestellt wird, muss ein Berechtichtigungssystem implementiert werden können.

4.4 Vorüberlegungen

Damit die Software im zeitlichen Rahmen der Bachelorarbeit entwickelt werden kann, muss eine Technologie eingesetzt werden, die eine effektive und effiziente Entwicklung ermöglicht. Für die Webentwicklung stehen verschiedene Programmiersprachen und entsprechend viele Technologien zur Verfügung. Weit verbreitet in der Webentwicklung sind die Programmiersprachen Java und PHP. Die Wahl fiel auf Python als Programmiersprache, da Python für das konkrete Projekt verschiedene Vorteile hat. Das zu integrierende »MHC-II binding prediction«-Programm ist zum Teil in Python programmiert, es kann direkt aus Python aufgerufen werden und die Implementierung der Schnittstelle wird wesentlich vereinfacht. Die Python-Standardbibliothek enthält eine API, die das Http-Protokoll implementiert. Für den Zugriff auf die NCBI-Proteindatenbank wird dadurch keine weitere Bibliothek benötigt.
Verschiedene Frameworks stehen für die Webentwicklung mit Python zur Verfügung. Für dieses Projekt wurde Django gewählt, da Django mehrere elementare Technologien wie eine Template Engine oder das objektrelationale Mapping vereint (im Abschnitt 3.2 wurden diese bereits erläutert). Die Dokumentation zu Django ist sehr ausgereift und online verfügbar,

was die Entwicklung enorm vereinfacht. Django hat sich über die Jahre etabliert, Fehler im Framework sind kaum vorhanden oder werden schnell korrigiert.
Die Entwicklung der Software muss unter Linux erfolgen, da das »MHC-II binding prediction«-Programm Teile enthält, die nur unter Linux lauffähig sind.
Die Erzeugung von Excel-Dateien kann mit der Python-Bibliothek Xlwt[2] implementiert werden. Mit der Bibliothek kann eine Excel-Datei mit verschiedene Formatierungen und mehreren »Sheets« erzeugt werden.
Für die Visualisierung und die Erzeugung von Grafiken wird die Python-Bibliothek PIL[3] verwendet. Die API bietet eine breite Auswahl an grafischen Elementen, die zum Zeichnen von Grafiken verwendet werden können. Gespeichert werden können die Grafiken in verschiedenen Formaten.
Als Datenbank wird das relationale Open-Source Datenbanksystem MySQL[4] eingesetzt. Für MySQL fallen keine Lizenzkosten an und es ist auf allen gängigen Linux-System verfügbar.

[2] Projekthomepage: http://www.python-excel.org
[3] http://www.pythonware.com/products/pil/
[4] http://www.mysql.com

5 Systemarchitektur

Im folgenden Abschnitt wird die Architektur der im Rahmen der Bachelorarbeit entwickelten Software erläutert, die fachlichen und die technischen Komponenten sowie das Datenbankmodell werden dargestellt.

5.1 Design der Anwendung

Das Anwendungsdesign definiert mehrere Komponenten, die über festgelegte Schnittstellen miteinander kommunizieren. Die Komponenten sind in fachliche und technische Komponenten unterschieden (vgl. Breu u. a. (2005), Notation vgl. Oestereich (2006)).

5.1.1 Fachliche Komponenten

In Abbildung 5.1 sind die fachlichen Komponenten der Webanwendung aufgeführt. Gesteuert werden alle Komponenten durch das Framework, das als zentrale Komponente in dem Diagramm dargestellt ist.
Die »Sequence Analyser«-Komponente lädt die Proteinsequenz aus der NCBI-Proteindatenbank und speichert diese in der lokalen Datenbank. Über eine Schnittstelle wird auf das »MHC-II binding prediction«-Programm zugegriffen, um für die einzelnen Peptide des Proteins die Bindungswahrscheinlichkeiten zu bestimmen. Die Bindungswahrscheinlichkeiten werden durch den »Sequence Analyser« visualisiert und dem Benutzer präsentiert. Mehrere Visualisierungsmöglichkeiten, die durch den Benutzer ausgewählt werden können, stehen zur Verfügung. Ein Export der Analyseergebnisse ist möglich.
Der »Sequence Analyser« stellt eine Schnittstelle bereit, über die andere Komponenten auf die Analyse- und Visualisierungs-Funktionalität zugreifen können. Mit der »Sequence Comparer«-Komponente können mehrere Proteinsequenzen in einem Schritt analysiert werden. Die Analyse wird durchgeführt indem auf den »Sequence Analyser« zugegriffen wird.
Der »Motif Finder« arbeitet in zwei Schritten: Im ersten Schritt wird auf das »MHC-II binding prediction«-Programm zugegriffen und eine Proteinsequenz auf gut bindende Peptide gefiltert, im zweiten Schritt werden die gut bindenden Peptide bezüglich der in Abschnitt

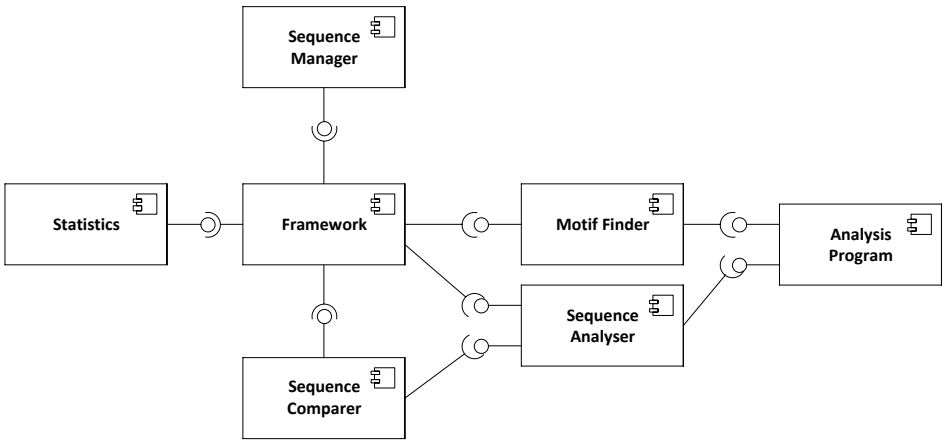


Abbildung 5.1: Fachliche Komponenten

2.2.3 beschriebenen SYFPEITHI Anker-Motive untersucht. Die Ergebnisse der Motivsuche werden in der Datenbank hinterlegt und können später erneut abgerufen, exportiert oder für statistische Auswertungen verwendet werden. Die Anker-Motive werden vom Benutzer angelegt und in der Datenbank gespeichert.

5.1.2 Technische Komponenten

In Abbildung 5.2 sind die technischen Komponenten der Software wiedergegeben.
Das in den fachlichen Komponenten allgemein als »Framework«-Komponente modellierte Framework wurde durch die »Django URL Dispatcher«-Komponente ersetzt. Diese leitet den Http-Request des Benutzers an die zuständige technische Komponente weiter. Die Komponente verarbeitet die Anfrage in einer View und liefert eine Http-Response zurück, die an den Browser des Benutzers zurückgegeben wird.
Die in den fachlichen Komponenten aufgeführte Komponente »Analysis Program« wurde durch eine Adapter-Komponente ersetzt, diese kapselt die Implementierungsdetails für den Zugriff auf das »MHC-II binding prediction«-Programm. Dadurch können beliebige Programme für die Berechnung der Bindungswahrscheinlichkeiten in die Software integriert werden.
Die in jeder technischen Komponente enthaltene Subkomponente »Views« implementiert Methoden, die durch das Django Framework aufgerufen werden. Diese wiederum rufen die jeweilige Subkomponente auf, um die Anwendungslogik der Komponente zu implementieren.

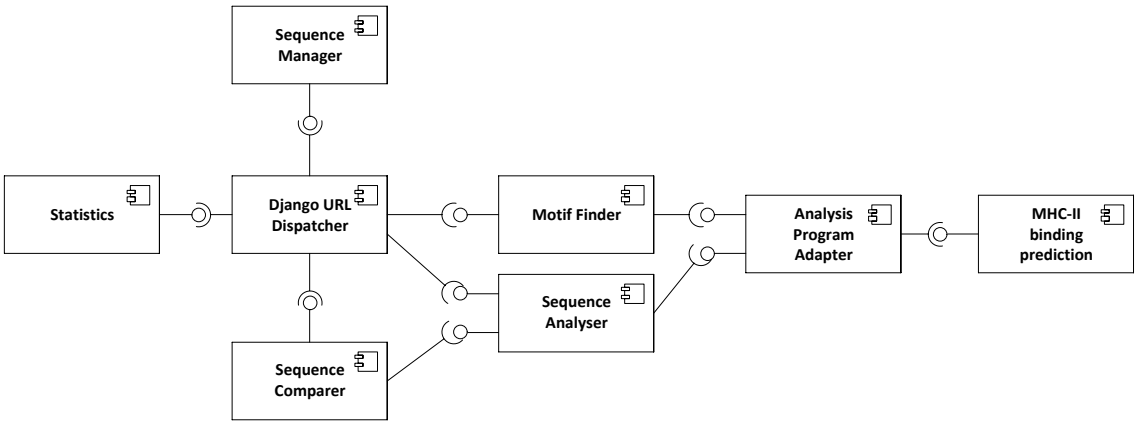


Abbildung 5.2: Technische Komponenten

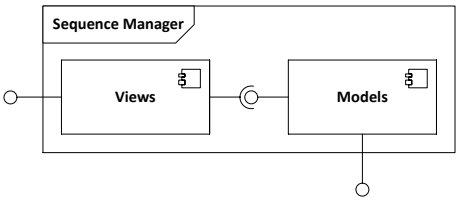


Abbildung 5.3: Subkomponenten der »Sequence Manager«-Komponente

»Sequence Manager«

Über die »Sequence Manager«-Komponente werden die in der Datenbank gespeicherten Proteine verwaltet. Diese sind in Kategorien und Unterkategorien organisiert. Abbildung 5.3 zeigt die in der »Sequence Manager«-Komponente enthaltenen Subkomponenten.
»Models« enthält die Klassen des objektrelationalen Mappings. Diese können von anderen Komponenten der Software verwendet werden, um Proteinsequenzen aus der Datenbank zu laden, anzulegen oder zu verändern.
Neben der Bereitstellung der Datenbankschnittstelle bietet der »Sequence Manager« eine Oberfläche, um die Proteinsequenzen in der Datenbank zu verwalten. Diese können über die Weboberfläche gelöscht oder für weitere Analysen mit den Modulen »Sequence Analyser«, »Sequence Comparer« und »Motif Finder« selektiert werden.

»Sequence Analyser«

In der Abbildung 5.4 sind die Subkomponten der »Sequence Analyser«-Komponente aufgeführt.
Für den Import aus der NCBI-Proteindatenbank steht die Komponente »NCBIDownload« zur Verfügung. Diese importiert das Datenblatt eines Proteins über das Http-Protokoll. Sofern der Download erfolgreich war, wird aus dem Datenblatt die Proteinsequenz und der Name des Proteins extrahiert und in der Datenbank gespeichert. Der Zugriff auf die Datenbank erfolgt mit der Komponente »Models« des »Sequence Managers«.
Die »Analyser«-Komponente greift über eine Schnittstelle auf die Adapter-Komponente des Programms zur Berechnung der Bindungswahrscheinlichkeit zu. Damit die Analyse durchgeführt werden kann, muss der Datensatz des Proteins vorher aus der Datenbank geladen werden. Der Analyser stellt seine Funktionalität über eine Schnittstelle auch anderen Komponenten zur Verfügung.
Für die Visualisierung der Analyseergebnisse werden die Komponenten »Image Visualisation« und »HTML Tabel Visualisation« verwendet. »Image Visualisation« erzeugt Grafiken, z.B. in Form eines Histogrammes. »HTML Table Visualisation« stellt die Ergebnisse in Tabellenform als HTML-Code dar.

»Sequence Comparer«

Mit dem »Sequence Comparer« (Abbildung 5.5) können mehrere in der Datenbank gespeicherte Proteinsequenzen in einem Schritt analysiert und verglichen werden.

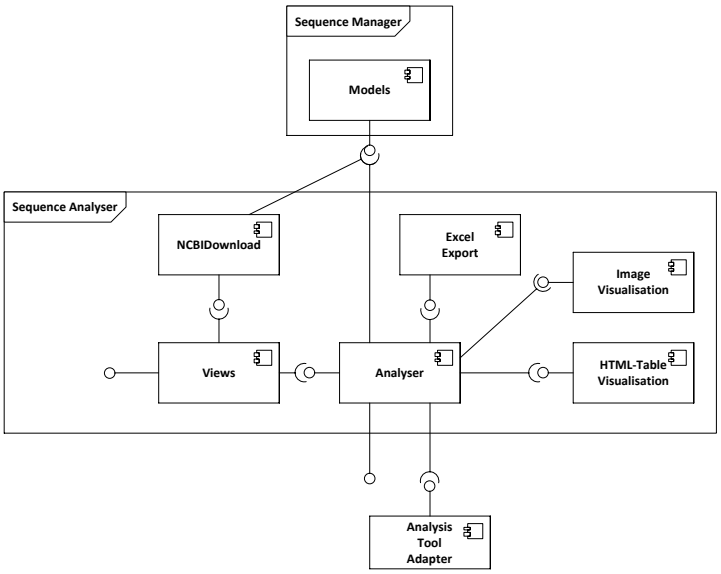


Abbildung 5.4: Subkomponenten der »Sequence Analyser«-Komponente

Der Zugriff auf die Datenbank erfolgt über die durch den »Sequence Manager« bereitgestellte »Models«-Komponente. Die Analyse wird durchgeführt, indem über eine Schnittstelle die »Analyser«-Komponete des »Sequence Analyser« aufgerufen wird.

»Motif Finder«

Die »Motif Finder«-Komponente (Abbildung 5.6) integriert die durch SYFPEITHI vorgegebenen Anker-Motive (siehe Abschnitt 2.2.3) in die Software.
Der »Finder« lädt Proteinsequenzen aus der Datenbank und bestimmt die Bindungswahrscheinlichkeiten für die einzelnen Peptide der Proteinsequenz. Der Zugriff auf die Datenbank erfolgt mit der »Models«-Komponente des »Sequence Managers«. Für die Durchführung der Analyse wird die »Analyser«-Komponente des »Sequence Analyser« verwendet. Die als gut bindend klassifizierten Peptide werden auf die Existenz der SYFPEITHI-Anker-Motive geprüft. Sind die Anker-Motive in dem Peptid vorhanden, werden sie in der Datenbank gespeichert. Hierfür werden die Klassen der »Models«-Komponente verwendet, die im »Motif Finder« integriert ist.
Bereits durchgeführte Analysen können über die »Result Browser«-Komponente erneut abgerufen werden, hierfür greift diese auf die eigene »Models«-Komponente zu, um die Analyseergebnisse aus der Datenbank zu laden.

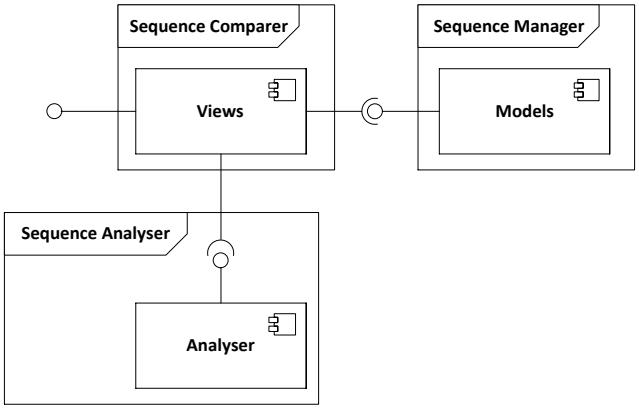


Abbildung 5.5: Subkomponenten der »Sequence Comparer«-Komponente

Das Verwalten, d.h. das Anlegen und Löschen der Anker-Motive in der Datenbank, erfolgt über die »Anchor Manager«-Komponente.

Integration des Analyse-Programms

Das »MHC-II binding prediction«-Programm vereint, wie bereits im Abschnitt 2.2.2 erläutert wurde, mehrere Programme. Die Programme implementieren verschiedene Algorithmen, um die Bindungswahrscheinlichkeiten für eine bestimmte Proteinsequenz zu berechnen. In Abbildung 5.7 ist das Programm als eigenständige Komponente dargestellt. Der Zugriff auf diese erfolgt über eine Adapter-Komponente, die die Implementierungsdetails abstrahiert. Somit wird es möglich, mit geringem Programmieraufwand das Analyseprogramm auszutauschen.

Statistik-Komponente

Die analysierten Proteinsequenzen werden mit der »Statistics«-Komponente statistisch ausgewertet, hier wird die Häufigkeitsverteilung der Aminosäuren innerhalb der Peptide untersucht.
Über diese Komponente kann eine in der Datenbank gespeicherte Analyse ausgewählt werden. Hierfür greift sie über die »Models« des »Sequence Managers« und des »Motif Finders« auf die Datenbank zu. Die durch den Motif-Finder gefilterten Peptide können selektiert

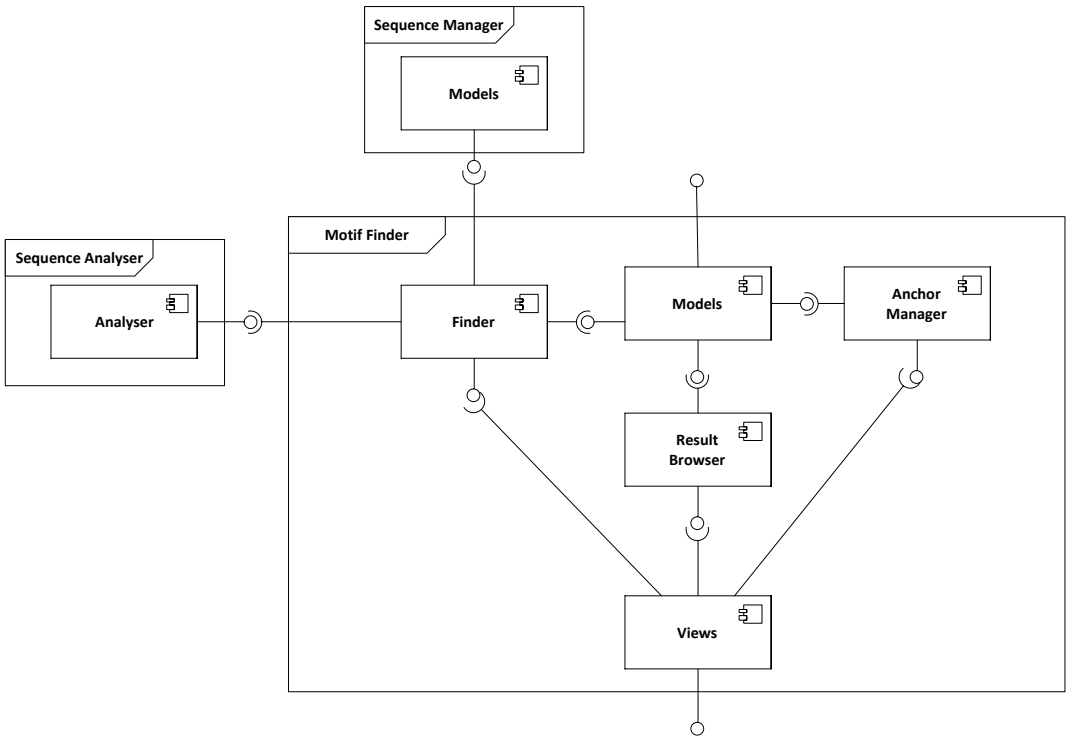


Abbildung 5.6: Subkomponenten der »Motif Finder«-Komponente

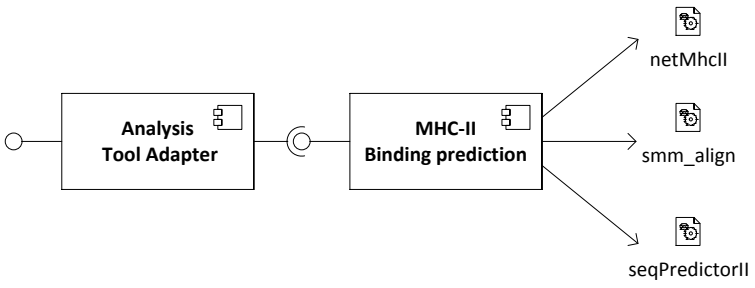


Abbildung 5.7: Integration des »MHC-II binding prediction«-Programms

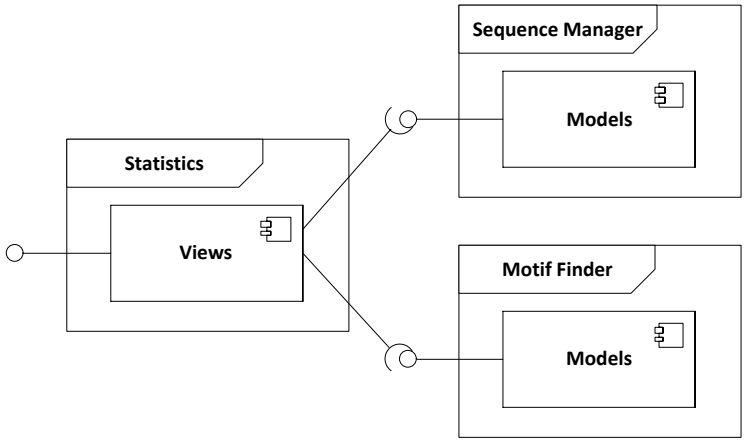


Abbildung 5.8: Statistik-Komponente

und statistisch untersucht werden. Dabei werden mehrere Analysen durchgeführt: Die Verteilung der Aminosäuren zwischen den Anker-Elementen, die Verteilung der Aminosäuren der Anker-Elemente und die Gesamtverteilung der Aminosäuren wird angezeigt. Desweiteren wird die Verteilung der Aminosäuren jeder einzelnen Position innerhalb des Peptids abgebildet.

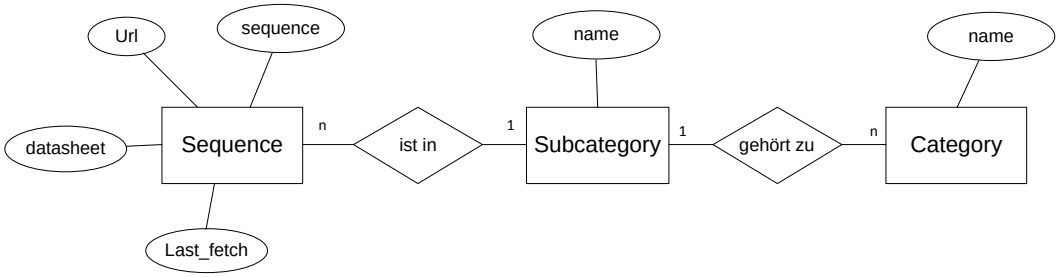


Abbildung 5.9: Entitäten der »Sequence Manager«-Komponente

5.2 Datenbankmodell

Der Zugriff auf die Datenbank erfolgt über das objektrelationale Mapping der Django-API. Die dazu notwendigen Klassen werden durch den »Sequence Manager« bereitgestellt. Der »Sequence Analyser«-Komponente verwendet diese Klassen, um die importierten Sequenzen mit Zuordnung zu einer Kategorie und Unterkategorie in der Datenbank zu speichern. Die »Motif Finder«-Komponente benötigt die Datenbank, um die Ergebnisse der Anker-Motivsuche und die zur Verfügung stehenden Anker-Motive zu speichern. Sie implementiert dafür eigene Datenbankklassen.
Die in diesem Abschnitt aufgeführten Abbildungen zeigen die Tabellen der Datenbank als »Entity Relation Model«-Diagramm in der Notation von Chen (vgl. Chen (1976)).

Speichern der Proteinsequenzen

Die Proteinsequenzen werden in den in Abbildung 5.9 aufgeführten Entitäten gespeichert. Die Entität »Sequence« speichert die aus der NCBI Proteindatenbank importierte Proteinsequenz in der Datenbank.

Hierbei werden folgende Daten gespeichert:

- »sequence«: Proteinsequenz in der Einbuchstaben-Kodierung.
- »datasheet«: Datenblatt des Proteins.
- »url«: URL, über die die Proteinsequenz gespeichert wurde.
- »last_fetch«: Der Zeitpunkt des letzten Imports

Jeder Proteinsequenz wird eine Kategorie und eine Unterkategorie zugeordnet. Die Kategorien unterscheiden zum Beispiel zwischen Bakterien und Viren. Die Unterkategorien der Viren-Kategorie können das EBV- oder Coxsackie-Virus sein. In der EBV-Virus Unterkategorie sind die Proteinsequenzen des EBV-Virus enthalten. Durch die Kategorisierung werden die Daten im »Sequence Manager« übersichtlicher dargestellt.
Die Namen der Kategorien und Subkategorien werden durch das Attribut »name« festgelegt. Die Einteilung eines Proteinsequenz-Datensatz erfolgt über einen Fremdschlüssel auf die Tabelle »Subcategory«. Diese wiederum enthält einen Fremdschlüssel auf die Tabelle »Category«, wodurch die Zuordnung zu einer Subkategorie und Kategorie zustande kommt.

Motif Finder

Die in Abbildung 5.10 aufgeführten Entitäten speichern die Ergebnisse der Motiv-Suche und die verfügbaren Anker-Motive der »Motif Finder«-Komponente in der Datenbank.

Anker-Motive

Die Anker-Motive werden in den Entitäten »Anchor« und »Anchorselector« gespeichert.
Die Entität »Anchor« repräsentiert ein Anker-Motiv, sie speichert den Namen des Ankers und den zugehörigen HLA-Typ. Dieser wird über einen Fremdschlüssel auf die Tabelle »HLA-Type« referenziert.
Die Entität »Anchorselector« enthält die zur Auswahl stehenden Aminosäuren eines Anker-Elements. Diese werden in der Tabelle »Acid« gespeichert, die über einen Fremdschlüssel referenziert wird.

Analyseergebnisse

Eine durch die »Motif Finder«-Komponente erstellte Analyse wird in der Entität »Analysation« gespeichert, wobei der Name der Analyse im Attribut »name« festgelegt werden kann. Desweiteren wird in den Attributen »min_threshold« und »max_threshold« ein Schwellenwert für die Bindungswahrscheinlichkeit gespeichert, anhand derer die Peptide für die Analyse

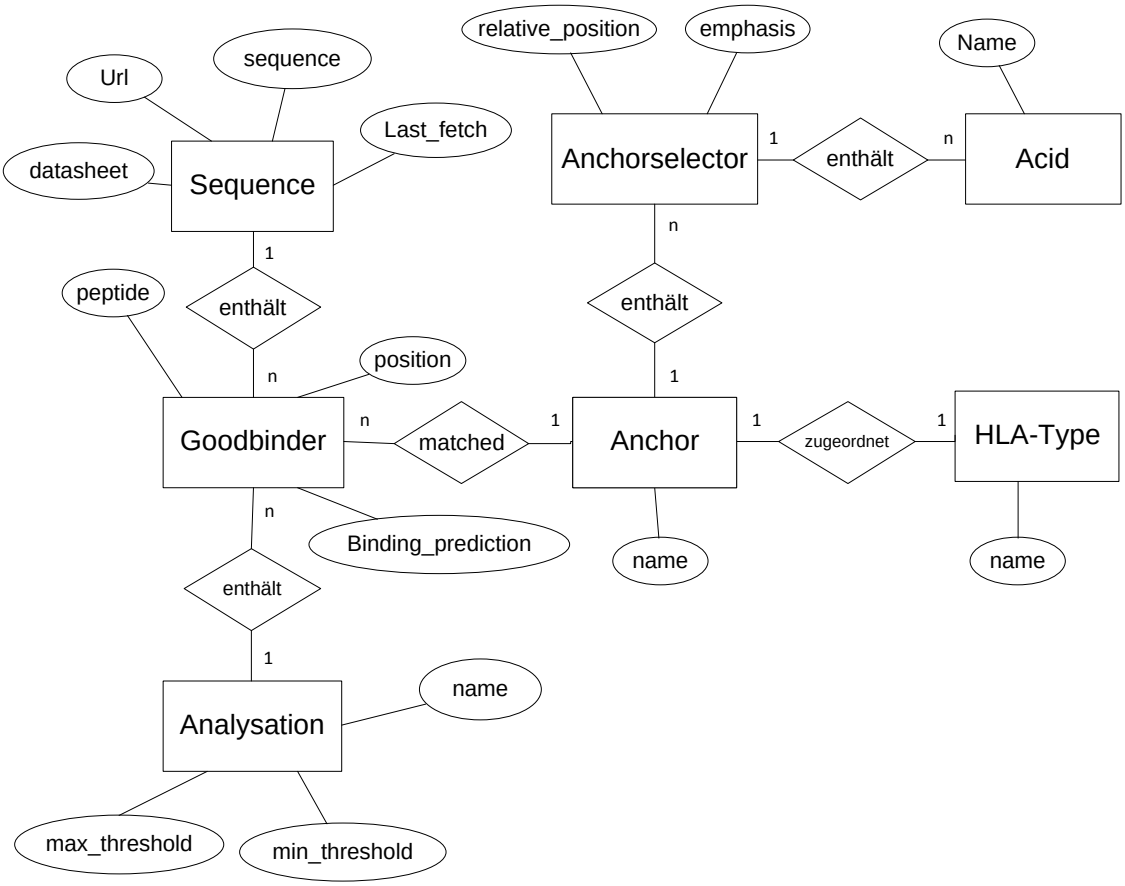


Abbildung 5.10: Entitäten der »Motif Finder«-Komponente

mit dem »Motif Finder« gefiltert werden. Diese Peptide werden in der Tabelle »Goodbinder« gespeichert, wobei die Analyse über einen Fremdschlüssel referenziert wird. Ebenfalls über einen Fremdschlüssel wird der zugehörige Proteinsequenz-Datensatz referenziert. Die Tabelle Goodbinder hat folgende Attribute:

- »peptide«: Peptid in der Einbuchstaben-Kodierung.
- »position«: Position des Peptids innerhalb der Proteinsequenz.
- »binding_prediction«: Bindungswahrscheinlichkeit als Zahlenwert.

6 Realisierung

Im folgenden Abschnitt wird im Detail auf die Umsetzung der einzelnen Komponenten eingegangen. Im ersten Teil des Abschnitts wird ein genereller Einblick in den Aufbau der Anwendung gegeben. Im weiteren Verlauf des Abschnitts wird im Detail die Umsetzung der einzelnen Komponenten erläutert.

6.1 Integration der Komponenten

MVC wurde durch die Django-Entwickler so interpretiert, dass der Controller das Framework realisiert und die Anwendungslogik der jeweiligen Komponente startet. In Django-Terminologie handelt es sich dabei um »Apps«, die die in Abschnitt 3.2 erläuterte Grundstruktur besitzen. Jede Komponente enthält mehrere Views, die durch das Framework aufgerufen werden. Diese Views implementieren die Anwendungslogik, d. h. sie erzeugen HTML, das an den Aufrufer zurückgegeben wird. Hierzu greifen sie auf die Subkomponenten zu, die in der jeweiligen Komponente enthalten sind.

6.2 Weboberfläche mit jQuery

Jede Webanwendung besteht aus der serverseitigen Anwendungslogik, einer clientseitigen Darstellung sowie der Interaktion mit dem Benutzer in Form des Browsers.
Die clientseitige Darstellung erfolgt durch HTML, Cascading Style Sheets (CSS) und JavaScript. HTML wird serverseitig erzeugt und an den Browser zur Darstellung ausgeliefert.
Bei der Programmierung mit Django werden Templates angelegt, die durch die Views mit Inhalt gefüllt werden. Jede Komponente der Software verfügt über eigene Templates, die Inhalte eines Master-Templates erben. Im Falle der hier entwickelten Anwendung enthält das Master-Template die Navigation und verschiedene Einbindungen von JavaScript- und CSS-Dateien. Abbildung 6.1 zeigt einen Ausschnitt der Weboberfläche des »Sequence Managers« im Browser Google Chrome. Im oberen Bereich ist die Navigation dargestellt, über die die einzelnen Komponenten der Software aufgerufen werden können. Bei der Entwicklung der Weboberfläche wurde an verschiedenen Stellen auf die JavaScript-Bibliothek jQuery und

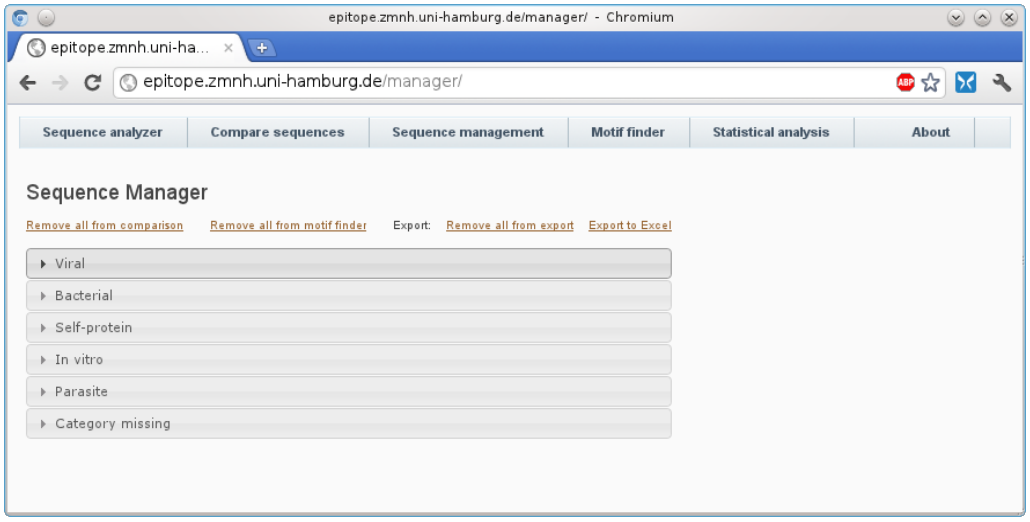


Abbildung 6.1: Oberfläche der »Sequence Manager«-Komponente

das Plugin jQueryUI zurückgegegriffen. Der Screenshot zeigt mehrere aufklappbare Reiter, die durch jQueryUI erzeugt werden.
Ein weiteres grafisches Bedienelement, das mit der jQueryUI-API implementiert wurde, ist das Dialog-Fenster, das innerhalb der Website angezeigt wird und beliebig vergrößer- und verschiebbar ist. Abbildung 6.2 zeigt einen Screenshot des für die Anzeige der Analyseergebnisse in der Statistik-Komponente verwendeteten Dialogfensters. Das Dialogfenster kann verwendet werden, um beliebige Inhalte anzuzeigen. Meist ist es sinnvoll, diese per Ajax vom Server vor dem Anzeigen des Dialogs nachzuladen. Der in der jQueryUI-API als grafisches Element vorhandene Fortschrittsbalken wurde ebenfalls an verschiedenen Stellen innerhalb der Anwendung verwendet.
Damit grafische Bedienelemente mit der jQueryUI-API erzeugt werden können, muss ein für die Visualisierung der Elemente verantwortliches Stylesheet eingebunden werden, das von den jQueryUI-Entwicklern zur Verfügung gestellt wird. Zum Erzeugen eines Dialogs muss ein HTML » «-Tag angelegt werden, das durch den jQueryUI-API-Befehl ».dialog()« zu einem Dialogfenster konvertiert wird. Im Anhang B ist ein Beispiel für das Programmieren mit jQueryUI aufgeführt.
Die Berechnung der Analysen benötigt je nach Auslastung des Servers und Umfang der Anfrage mehrere Sekunden. Damit die Benutzeroberfläche bedienbar bleibt, werden die Anfragen per Ajax an den Server gesendet. Für den Benutzer erfolgt das transparent im Hintergrund, die Aktivität wird durch einen Fortschrittsbalken signalisiert (siehe Abbildung 6.3). Die Webanwendung bleibt während der Anfrage und der Berechnung durch den Server benutzbar.

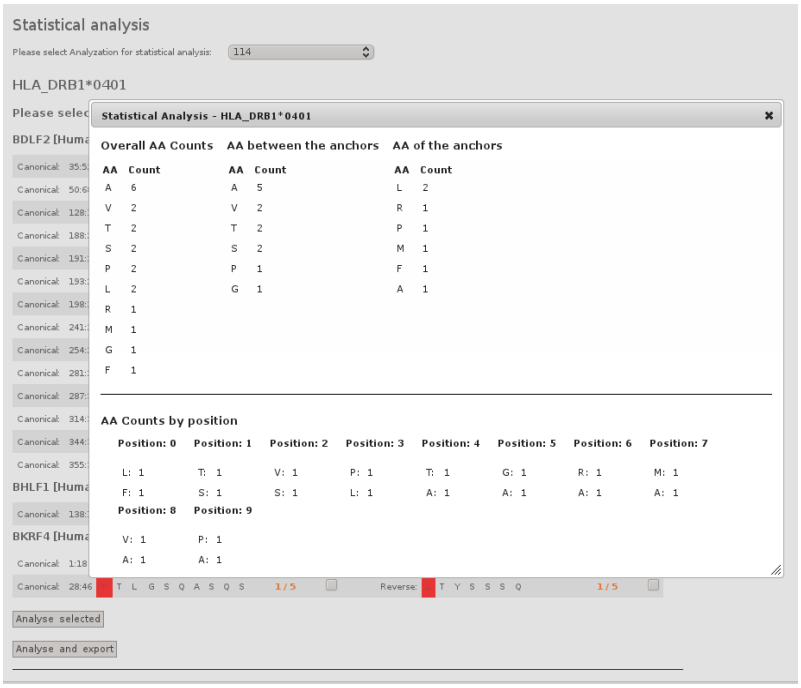


Abbildung 6.2: Dialogfenster der Statistik-Komponente

Sobald die Analyse fertiggestellt wurde, wird die Antwort des Servers zum HTML-Code hinzugefügt und dem Benutzer angezeigt.

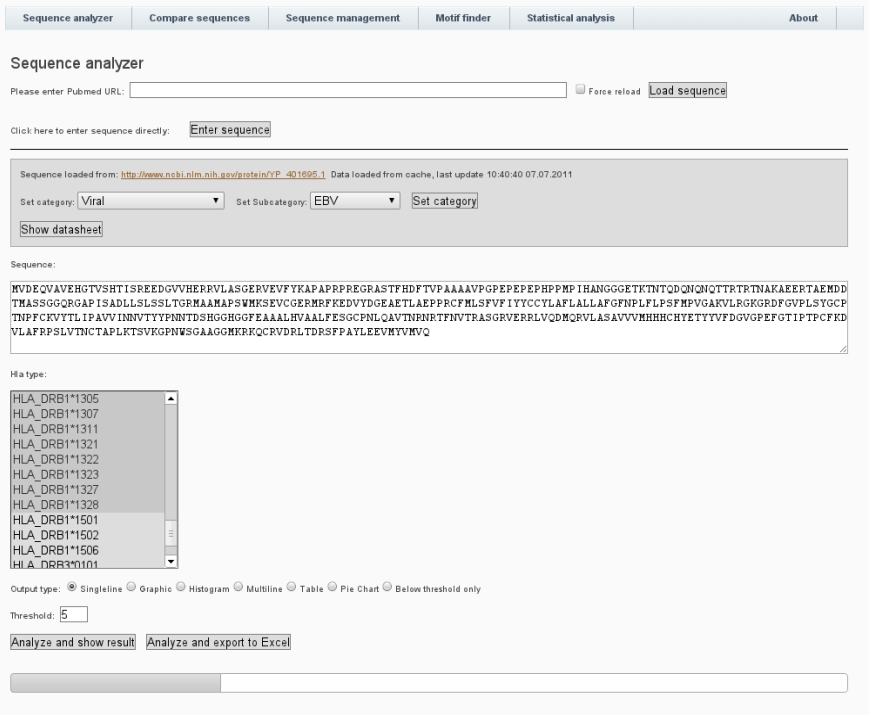


Abbildung 6.3: Analyse wird durchgeführt

Abbildung 6.4: Ergebnis der Analyse

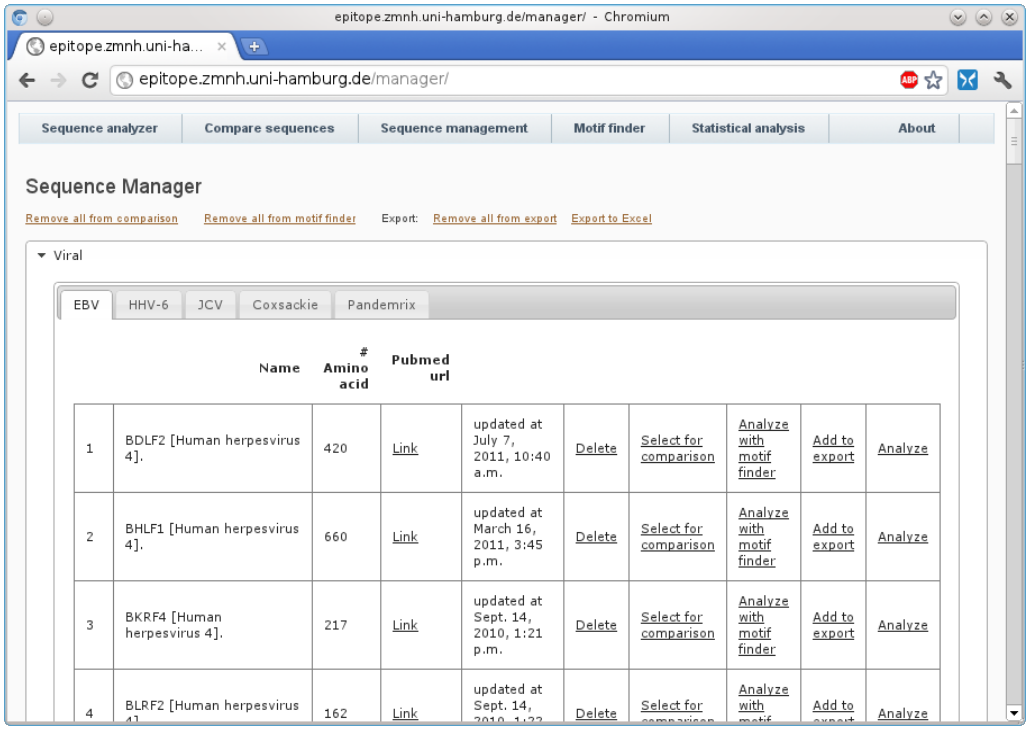


Abbildung 6.5: Benutzeroberfläche der »Sequence Manager«-Komponente

6.3 Verwalten der Proteinsequenzen

Die »Sequence Manager«-Komponente verwaltet die in der Datenbank gespeicherten Proteinsequenzen. Über den »Sequence Manager« können die Kategorien und Unterkategorien der Proteine aufgerufen werden. Die anderen Komponenten des Systems (z.B. »Sequence Analyser«) benutzen den »Sequence Manager« als Persistenzschicht, um Datenbankobjekte zu instanziieren oder zu speichern. Abbildung 6.5 zeigt einen Screenshot der Bedienoberfläche.
Für die Programmierung wurden zwei Benutzeroberflächen-Elemente von jQueryUI verwendet. Die Kategorien werden über das aufklappbare »Accordion«-Widget angezeigt (siehe Abbildung 6.6). Die Unterkategorien werden in den als »Tabs« bezeichneten Karteireitern dargestellt (siehe Abbildung 6.7). Damit der Seitenaufbau in einer möglichst kurzen Zeit erfolgt, werden die Inhalte des »Accordion« und der »Tabs« beim Öffnen einer Kategorie per Ajax nachgeladen. Hierdurch wird immer nur der dargestellte Teil der Kategorien und Unterkategorien beim Aufruf des »Sequence Managers« ausgeliefert.
Neben der Funktionalität zum Anzeigen der Kategorien und Unterkategorien können Daten-

Abbildung 6.6: Geschlossene Kategorien des »Sequence Managers«

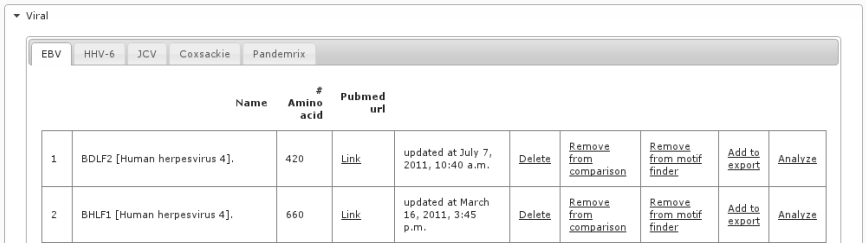


Abbildung 6.7: Geöffnete Kategorie »EBV« des »Sequence Managers«

sätze gelöscht oder für die Analyse mit den anderen Komponenten (z.B. »Motif-Finder oder »Sequence Comparer«) ausgewählt werden. Die Auswahl mehrerer Proteinsequenzen wird per Ajax an den Server gesendet und serverseitig in der Benutzer-»Session« gespeichert.
Die Datensätze des »Sequence Managers« können in das MS-Excel-Format exportiert werden, indem die zu exportierenden Datensätze selektiert werden. Über einen Export-Link wird eine Excel-Datei mit den Datensätzen erzeugt und an den Benutzer gesendet.
Der »Sequence Manager« ist die zentrale Komponente der Anwendung, zum Starten einer Analyse werden die zu analysierenden Datensätze im »Sequence Manager« ausgewählt.

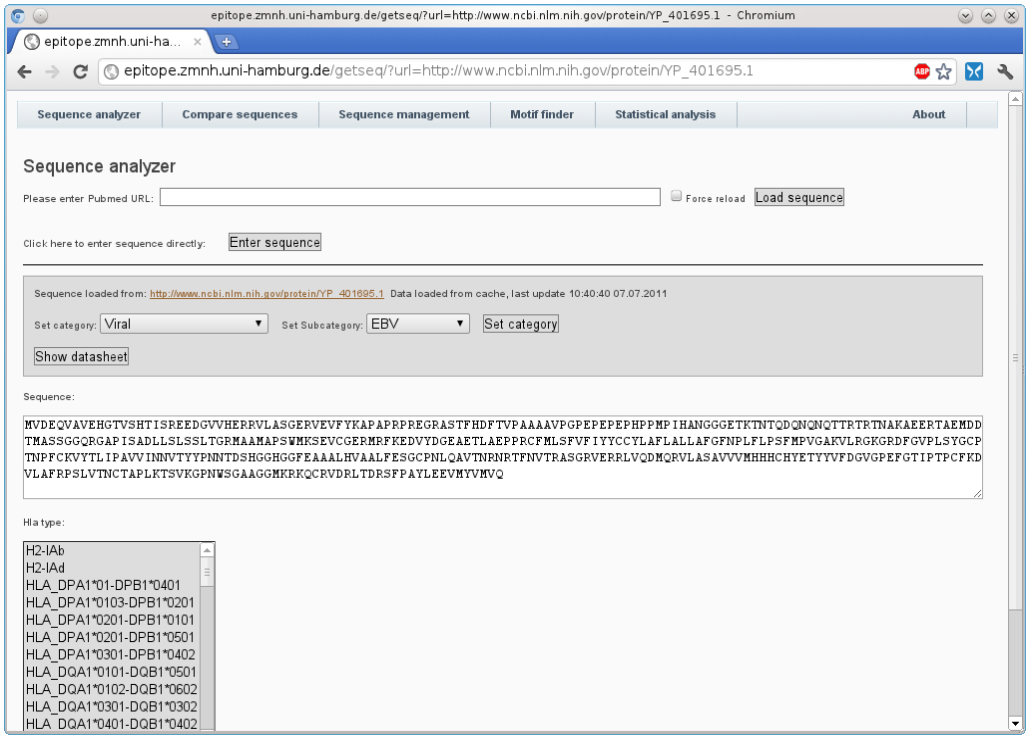


Abbildung 6.8: Benutzeroberfläche des »Sequence Analysers«

6.4 Die Analyse- und Visualisierungskomponente

Die Analyse- und Visualisierungskomponente ist im Komponentendiagramm als »Sequence Analyser«-Komponente aufgeführt, Abbildung 6.8 zeigt die Benutzeroberfläche.
Die Komponente enthält mehrere Views, die durch Zugriff auf die Sub-Komponenten die Anwendungslogik implementieren. Die Views werden entweder direkt durch Benutzer oder per Ajax aufgerufen.

Zugriff auf das Analyseprogramm

Eine nichtfunktionale Anforderung an die Software ist, dass das Analyseprogramm möglichst einfach ersetzt werden kann (siehe Abschnitt 4.3). Um das zu ermöglichen, wurde eine Adapter-Komponente entwickelt, die die Implementierungsdetails des Analyseprogrammes kapselt. Diese wurde in Form einer Python-Klasse implementiert, ihre Signatur ist im Listing 6.1 vereinfacht dargestellt.
Die Methode »do_analysis()« ruft das Analyseprogramm auf und gibt eine festgelegte Datenstruktur zurück. Als Parameter nimmt sie den HLA-Typ und eine Proteinsequenz entgegen. Sie berechnet die Bindungswahrscheinlichkeiten für die Peptide der Proteinsequenz und das als HLA-Typ angegebene MHC-II-Molekül.

```
1 class AnalysationAdapter:
2     def do_analysis(hla, sequence):
3         [...]
4         return analysis_results
```

Listing 6.1: Adapter-Klasse für den Zugriff auf das Analyseprogramm

Die zurückgelieferte Datenstruktur ist in Listing 6.2 gekürzt dargestellt, dabei handelt es sich um ein Array dreistelliger Tupel. Jedes Tupel enthält als erstes Element die Position des Peptids innerhalb der Proteinsequenz. Das zweite Element ist die Peptidsequenz für das die Bindungswahrscheinlichkeit bestimmt wurde. Im dritten Element des Tupels ist die berechnetete Bindungswahrscheinlichkeit festgelegt. Ein kleiner Wert stellt eine hohe Bindungswahrscheinlichkeit dar.

```
1 analysis_results = [
2     ('113:127', 'PQRSPRTRQAGYALG', 34.99),
3     ('114:128', 'QRSPRTRQAGYALGE', 15.12),
4     ('115:129', 'RSPRTRQAGYALGEG', 7.35),
5     # [..]
6 ]
```

Listing 6.2: Von der »do_analysis«-Methode zurückgegebene Datenstruktur

Der Zugriff der Adapterklasse auf das »MHC-II binding prediction«-Programm erfolgt über ein Python-Script. Das Python-Script kann sowohl als Kommandozeilenprogramm, als auch als CGI-Script zum Einbinden in den Webserver verwendet werden. Die Architektur des Analyseprogramms ist in Abbildung 6.9 vereinfacht dargestellt.
Das Python-Skript startet die verschiedenen Programme zur Berechnung der Bindungswahrscheinlichkeiten (Abschnitt 2.2.2). Die Ergebnisse werden als HTML oder als Textausgabe

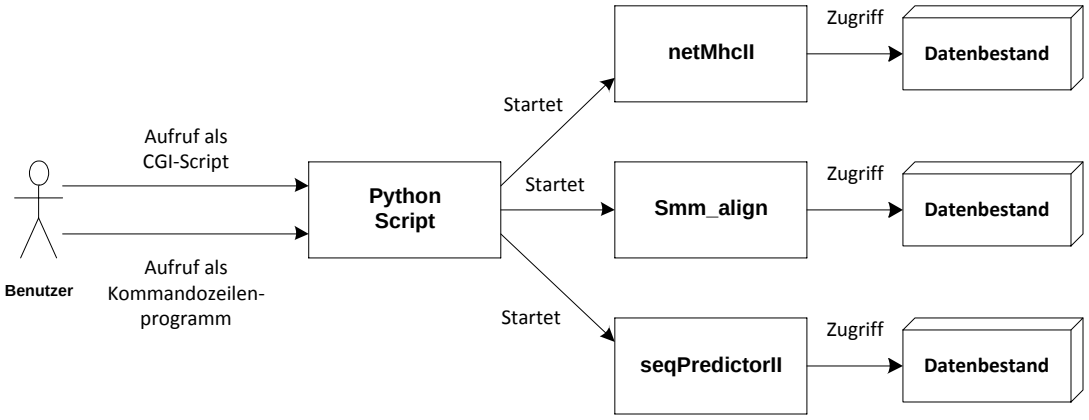


Abbildung 6.9: Architektur des »MHC-II binding prediction«-Programm

zurückgegeben. Dabei wird unterschieden, ob der Aufruf als CGI-Skript oder Kommmandozeilenprogramm erfolgte.
In dem Skript sind mehrere Klassen enthalten, in denen die Analysealgorithmen implementiert sind. Die Adapterklasse importiert die Klassen, setzt verschiedene Parameter und startet die Analyse. Hierfür wird die Klasse »MHCBindingPrediction« verwendet, die Klassenmethode »predict()« startet die Berechnung und liefert das Ergebnis, das durch die Adapterklasse in die festgelegte Datenstruktur (Listing 6.2) umgewandelt wird.

Import einer Sequenz aus der NCBI-Proteindatenbank

Für den Import einer Proteinsequenz aus der NCBI-Proteindatenbank wurde die Komponente »NCBIDownload« entwickelt. Eine Sequenz kann unter Angabe eines Links zu einem Datenblatt in der NCBI-Proteindatenbank importiert werden. Hierzu ruft die Komponente per Http-Request den Link auf (dieser kann zum Beispiel »http://www.ncbi.nlm.nih.gov/protein/YP_401695.1« sein) und lädt den vom Server zurückgegeben HTML-Code.
In den Metadaten des HTML-Dokuments ist der Eintrag »ncbi_uidlist« enthalten, der eine eindeutige Identifikation des Datensatzes in der Datenbank enthält. Die dort enthaltene ID wird verwendet, um über die Export-Schnittstelle der NCBI-Proteindatenbank das Datenblatt als Textdatei herunterzuladen. Die in dem Datenblatt als Zeichenkette vorhandene Proteinsequenz wird extrahiert und in der Datenbank gespeichert.

HLA_DRB1*0901

Allele	Position	Sequence	Consensus percentile
HLA DRB1*0901	1:1-15	LGSRGPRPHPAFQVQ	29.45
HLA DRB1*0901	1:2-16	GSRGPRPHPAFQVQW	31.24
HLA DRB1*0901	1:3-17	SRGPRPHPAFQVQWS	31.34
HLA DRB1*0901	1:4-18	RGPRPHPAFQVQWSA	31.34
HLA DRB1*0901	1:5-19	GPRPHPAFQVQWSAR	53.40
HLA DRB1*0901	1:6-20	PRPHPAFQVQWSARN	31.34
HLA DRB1*0901	1:7-21	RPHPAFQVQWSARNP	32.63
HLA DRB1*0901	1:8-22	PHPAFQVQWSARNPG	30.73
HLA DRB1*0901	1:9-23	HPAFQVQWSARNPGC	30.00
HLA DRB1*0901	1:10-24	PAFQVQWSARNPGCP	4.94
HLA DRB1*0901	1:11-25	AFQVQWSARNPGCPR	12.90
HLA DRB1*0901	1:12-26	FQVQWSARNPGCPRT	13.18
HLA DRB1*0901	1:13-27	QVQWSARNPGCPRTW	32.81
HLA DRB1*0901	1:14-28	VQWSARNPGCPRTWR	33.45

Abbildung 6.10: »Table«-Visualisierung der Analyseergebnisse

Visualisierung der Analyseergebnisse

Die Visualisierung wird durch die »Analyser«-Komponente gesteuert, indem auf die »Image Visualisation«- und »HTML-Table Visualisation«-Komponenten zugegriffen wird. Zur Auswahl stehen insgesamt sieben verschiedene Darstellungsmöglichkeiten. Unterschieden wird zwischen tabellarischen Visualisierungen, die HTML Code erzeugen, und grafischen Visualisierungen, die Grafiken generieren.

»Table«-Visualisierung

Die in Abbildung 6.10 dargestellte »Table«-Visualisierung ist eine einfache Darstellung der durch das Analyseprogramm zurückgegebenen Datenstruktur.
Die einzelnen Spalten enthalten den HLA-Typ, die Position des Peptids innerhalb der Proteinsequenz und die Peptidsequenz. In der letzten Spalte ist die Bindungswahrscheinlichkeit dargestellt.

»Singleline«-Visualisierung

Abbildung 6.11 zeigt die Analyseergebnisse als »Singleline«-Visualisierung. Die Abbildung enthält den HLA-Typ, in dem Beispiel HLA_DRB*0901, für den die Bindungswahrscheinlichkeit bestimmt wurde. Die drei Zeilen »# below threshold«, »Percent below threshold« und »Average percentage« sind statistische Auswertungen. »# below threshold« gibt die Anzahl der Peptide in der Sequenz wieder, deren Bindungswahrscheinlichkeit unter einem bestimmten, durch den Benutzer festgelegten Schwellenwert (Threshold) liegt. »Percent below thres-

hold« ist der prozentuale Anteil der Peptide, deren Bindungswahrscheinlichkeit unterhalb des Schwellenwertes liegt. »Average percentage« repräsentiert die durchschnittliche Bindungswahrscheinlichkeit.

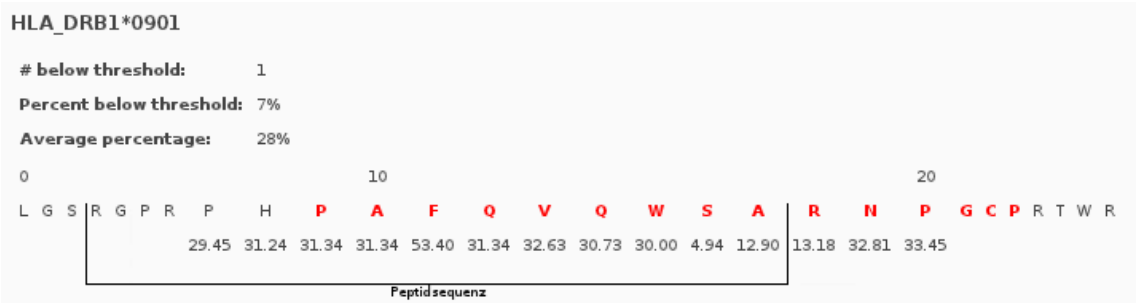


Abbildung 6.11: »Singleline«-Visualisierung der Analyseergebnisse

Die drei folgenden Zeilen zeigen die Peptidsequenz und die Bindungswahrscheinlichkeiten der einzelnen Peptide. Die mit 0 beginnende Zeile markiert die Position innerhalb der Proteinsequenz, in dem Beispiel befindet sich die Aminosäure A an Position 10.
Die folgende Zeile zeigt die einzelnen Aminosäuren der Proteinsequenz. Die Aminosäuren der gut bindenden Peptide sind rot hervorgehoben. Die dritte Zeile enthält die einzelnen Bindungswahrscheinlichkeiten der Peptide. Diese beziehen sich immer auf das direkt über dem Wert aufgeführte Peptid, das eine Länge von 15 Aminosäuren hat. An Position 10 ist die Aminosäure A (Alanin) aufgeführt, unter der die Bindungswahrscheinlichkeit 31.34 aufgeführt ist. Die sieben Aminosäuren links und rechts von Alanin bilden das 15er Peptid, für die die Bindungswahrscheinlichkeit gilt. Aus der Grafik geht hervor, dass das Peptid »RGPRPHPAFQVQWSA« eine Bindungswahrscheinlichkeit von 31.34 für das »HLA_DRB1*0901« MHC-II Molekül besitzt.

»Multiline«-Visualisierung

Die Multiline Darstellung (Abbildung 6.12) ist eine Abwandlung der Singleline-Darstellung. Pro Peptid sind insgesamt drei Zeilen aufgeführt. Die erste Zeile zeigt die Position innerhalb des gesamten Proteins, die zweite Zeile zeigt die Proteinsequenz, wobei das Peptid hervorgehoben ist. Falls die Bindungswahrscheinlichkeit unter dem »Threshold« liegt, wird dieses rot hervorgehoben. In der dritten Zeile ist die Bindungswahrscheinlichkeit aufgeführt.

»Below threshold«-Visualisierung

Die »Below threshold«-Visualisierung (Abbildung 6.13) ist eine tabellarische Aufführung der Peptide, deren Bindungswahrscheinlichkeiten einen bestimmten Wert unterschreiten.

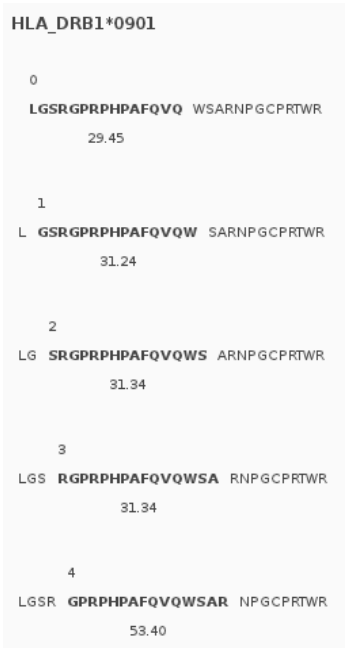
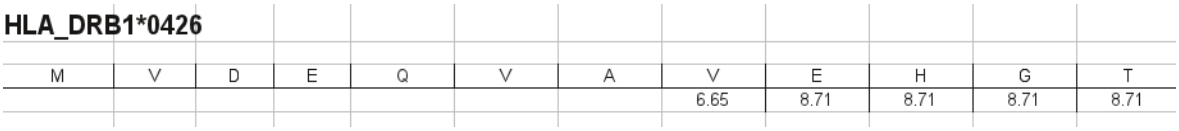


Abbildung 6.12: »Multiline«-Darstellung der Analyseergebnisse

Abbildung 6.13: »Below threshold«-Visualisierung



HLA_DRB1*0426

M	V	D	E	Q	V	A	V	E	H	G	T
							6.65	8.71	8.71	8.71	8.71

Abbildung 6.14: Excel-Export der »Singleline«-Visualisierung

Export der Analyseergebnisse

Die zuvor genannten tabellarischen Visualisierungen können in das Microsoft-Excel-Format übertragen werden. Hierfür wird durch die im »Sequence-Analyser« vorhandende »Excel-Export«-Komponente eine Excel-Datei erzeugt.
Python stellt dafür die Open-Source Bibliothek »xlwt«[1] zur Verfügung, die Excel-Dokumente mit mehreren Sheets und Formatierungen generieren kann.
In Abbildung 6.14 ist ein Excel-Export der »Singleline«-Visualisierung gezeigt. Im Anhang A sind weitere Screenshots der Exportfunktion enthalten.

Grafische Visualisierungen

Singleline als Grafik

Abbildung 6.15 zeigt die grafische Darstellung der Singleline-Visualisierung. Die Bindungswahrscheinlichkeiten sind auf der Y-Achse doppelt logarithmisch aufgetragen, die X-Achse enthält die Aminosäuren. Durch die doppelt logarithmische Darstellung werden niedrige Bindungswahrscheinlichkeiten als hohe Balken dargestellt und umgekehrt. Das ist gewünscht, da ein kleiner Wert eine hohe Bindungswahrscheinlichkeit bedeutet. Die rot hervorgehobenen Balken zeigen Werte, die den festgelegten Schwellenwert (Threshold) unterschreiten.

Verteilung der Bindungswahrscheinlichkeiten

Die in Abbildung 6.16 enthaltene Grafik zeigt die Verteilung der Bindungswahrscheinlichkeiten. Auf der X-Achse sind die abgerundeteten Werte für die Bindungswahrscheinlichkeiten eingetragen, 0 beinhaltet alle Werte von 0 - 0.999. In Y-Richtung sind durch die Balken die Anzahl der Peptide mit der auf der X-Achse aufgeführten Bindungswahrscheinlichkeit dargestellt. In dem Beispiel haben ungefähr 30 Peptide eine Bindungswahrscheinlichkeit von 3.

[1] http://www.python-excel.org

Abbildung 6.15: »Singleline«-Visualisierung als Balkengrafik

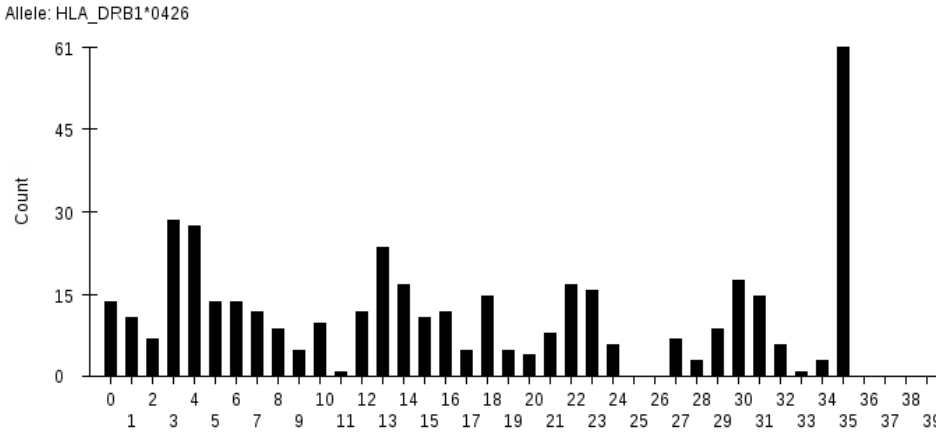


Abbildung 6.16: Verteilung der Bindungswahrscheinlichkeiten

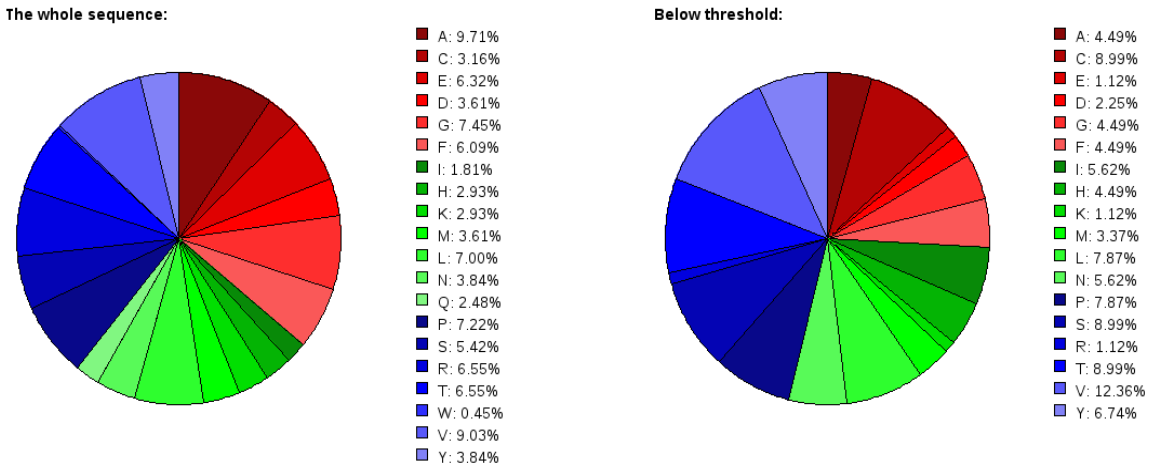


Abbildung 6.17: Häufigkeitsverteilung der Aminosäuren

Verteilung der Aminosäuren

Die Häufigkeitsverteilung der Aminosäuren wird in zwei Tortendiagrammen dargestellt (Abbildung 6.17).
Das erste Tortendiagramm zeigt die Häufigkeitsverteilung der Aminosäuren in der gesamten Proteinsequenz. Das zweite Tortendiagramm zeigt die Häufigkeitsverteilung der Aminosäuren, die in Peptidsequenzen vorhanden sind und deren Bindungswahrscheinlichkeit unterhalb eines bestimmten Schwellenwertes liegt.
In der Legende sind die relativen Häufigkeiten der einzelnen Aminosäuren als Prozentzahlen aufgeführt.

6.5 Analyse mehrerer Proteinsequenzen

Zur Analyse mehrerer Proteinsequenzen wurde die »Sequence Comparer«-Komponente entwickelt. Der Benutzer wählt in der Verwaltungsoberfläche (Abschnitt 6.3) mehrere Proteinsequenzen aus, die analysiert werden sollen. Die Ausgabe wird anhand des HLA-Typs gruppiert und dem Benutzer präsentiert. Abbildung 6.18 zeigt den »Sequenze Comparer« mit zwei ausgewählten Proteinsequenzen.
Der »Sequence Comparer« wird verwendet, um die Bindungseigenschaften mehrerer Proteinsequenzen zu vergleichen.

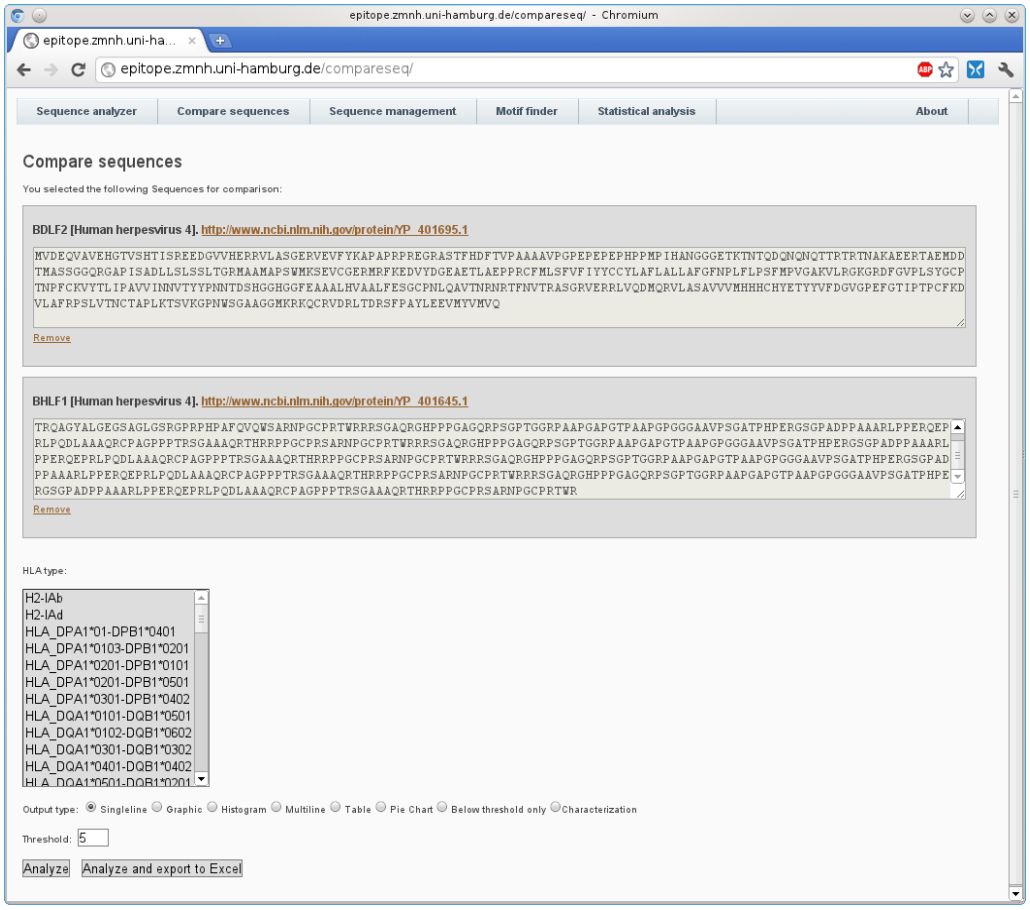


Abbildung 6.18: Bedienoberfläche der »Sequence Comparer«-Komponente

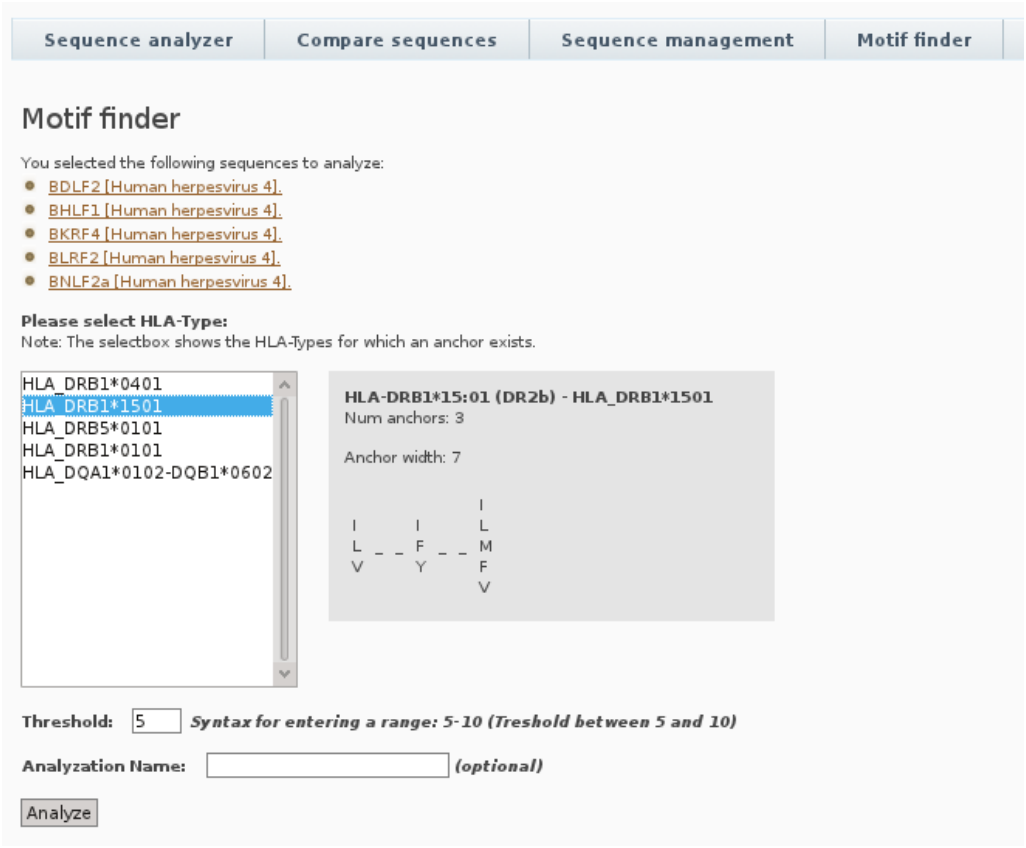


Abbildung 6.19: Benutzeroberfläche der »Motif Finder«-Komponente

6.6 Realisierung der »Motif Finder«-Komponente

Über den »Motif Finder« werden gut bindende Peptidsequenzen auf das Vorkommen von SYFPEITHI-Ankermotiven (Abschnit 2.2.3) untersucht.
Abbildung 6.19 zeigt die Benutzeroberfläche des »Motif Finders« in der mehrere Proteinsequenzen für die Analyse ausgewählt wurden. Die Anker-Motive sind spezifisch für den HLA-Typ und wurden in der Regel für das zugehörige MHC-II-Molekül experimentell ermittelt (siehe Abschnitt 2.2.3). Die Ergebnisse der Motivsuche sind in Abbildung 6.20 aufgeführt.

Implementierung im Detail

Die Anker-Motive werden durch die in Abbildung 6.21 vereinfacht dargestellte Klasse »Motif« implementiert. Dafür bilden mehrere Objekte der Klasse eine verkettete Liste, die den gesamten Anker enthält. Dieser wird anhand der Daten aus der Datenbank erzeugt.

BDLF2 [Human herpesvirus 4].

Position in Sequence	Binding prediction	Anchor Position	Number of Matching		Matching Canonical							
33:50	4.42%	38	2 / 3	SGER	V	E	V	F	Y	K	A	APRPRE
131:149	4.18%	134	1 / 3	AP	I	S	A	D	L	L	S	SSLTGRMAA
180:195	2.80%	191	2 / 3	EPPRCFMLSF	V	F	I	Y	Y	C		
185:214	2.01%	191	2 / 3	FMLSF	V	F	I	Y	Y	C	C	LAFLALLAFGFNPLFLP
203:217	4.97%	204	2 / 3		L	L	A	F	G	F	N	LFLPSFM
210:225	3.30%	214	3 / 3	PLF	L	P	S	F	M	P	V	AKVLR
315:333	4.22%	317	1 / 3	R	L	V	Q	D	M	Q	R	LASAVVVMHH
328:343	3.65%	329	1 / 3		V	V	V	M	H	H	H	HYETYYVF
355:375	2.01%	362	2 / 3	TPCFKD	V	L	A	F	R	P	S	VTNCTAP

Abbildung 6.20: Darstellung der Ergebnisse der Anker-Motivsuche

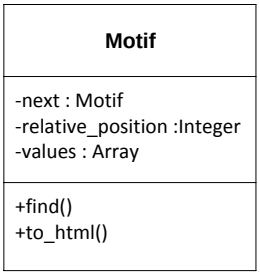


Abbildung 6.21: Vereinfachtes Klassendiagramm der Motif-Klasse

Die möglichen Aminosäuren jedes Anker-Elements werden durch die Klassenvariable »values« als Array gespeichert. Die verkettete Liste wird über das Attribut »next« erzeugt, das das nächste Element referenziert. Jedes Anker-Element besitzt einen Abstand zum vorherigen Element, der durch den Integer-Wert »relative_position« festgelegt wird.
Um zu prüfen, ob ein Anker auf eine Peptidsequenz passt, wird die Methode »find()« aufgerufen. Diese liefert »True« zurück, falls der Anker passt und »False«, falls er nicht passt. Die »find()«-Methode wird rekursiv in der verketteten Liste aufgerufen, wobei die Peptidsequenz weitergereicht wird.
Die Methode »to_html()« stellt die Peptidsequenz als HTML dar und hebt die Anker hervor.

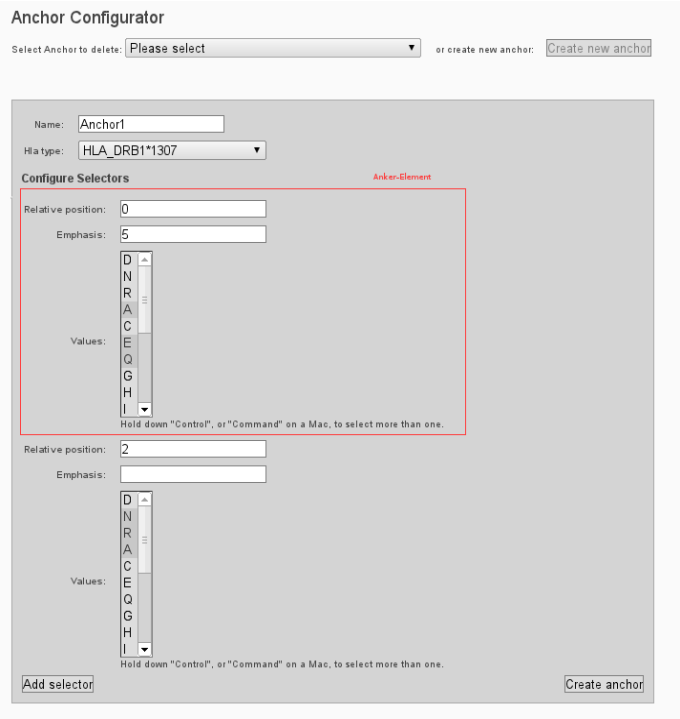


Abbildung 6.22: Anlegen eines Anker-Motivs

Verwaltung der Anker-Motive

Abbildung 6.22 zeigt die Verwaltungsoberfläche für die Anker-Motive. Über die »Selectbox« können Anker-Motive zum Löschen ausgewählt werden.
Durch Ausfüllen des grau hinterlegten Formulars wird ein Anker-Motiv angelegt. Hierzu wird ein Name eingegeben und der zugehörige HLA-Typ ausgewählt. Jedes Anker-Element wird

durch die drei Formularfelder »Relative position«, »Emphasis« und »Values« festgelegt (in der Abbildung durch einen roten Kasten hervorgehoben). »Relative position« legt den Abstand zum vorherigen Anker-Element fest. »Emphasis« ist ein optionaler Wert, der eine Gewichtung des Anker-Elements festlegt. Dieser kann später für statistische Auswertungen verwendet werden, in der aktuellen Implementierung wird er nicht benutzt. Über die Auswahl von »Values« können die Aminosäuren des Anker-Elements festgelegt werden.
Zur Eingabe weiterer Anker können zusätzliche Formulare durch die Schaltfläche »Add selector« hinzugefügt werden.

6.7 Statistische Analyse

In der »Statistics«-Komponente kann der Benutzer eine Analyse auswählen, zu der alle zugeordneten Peptide angezeigt werden. Aus diesen Peptiden kann eine Auswahl zur statistischen Analyse getroffen werden. Abbildung 6.23 zeigt die Oberfläche zum Auswählen der Peptide.
Die Ergebnisse der statistischen Analyse werden in einem Dialogfenster gezeigt (siehe Abbildung 6.24). Es werden insgesamt vier verschiedene statistische Analysen durchgeführt: Verteilung der Aminosäuren über die gesamte Sequenz, Verteilung der Aminosäuren zwischen den Anker-Elementen, Verteilung der Aminosäuren in den Anker-Elementen und die Verteilung der Aminosäuren an bestimmten Positionen der Sequenz.

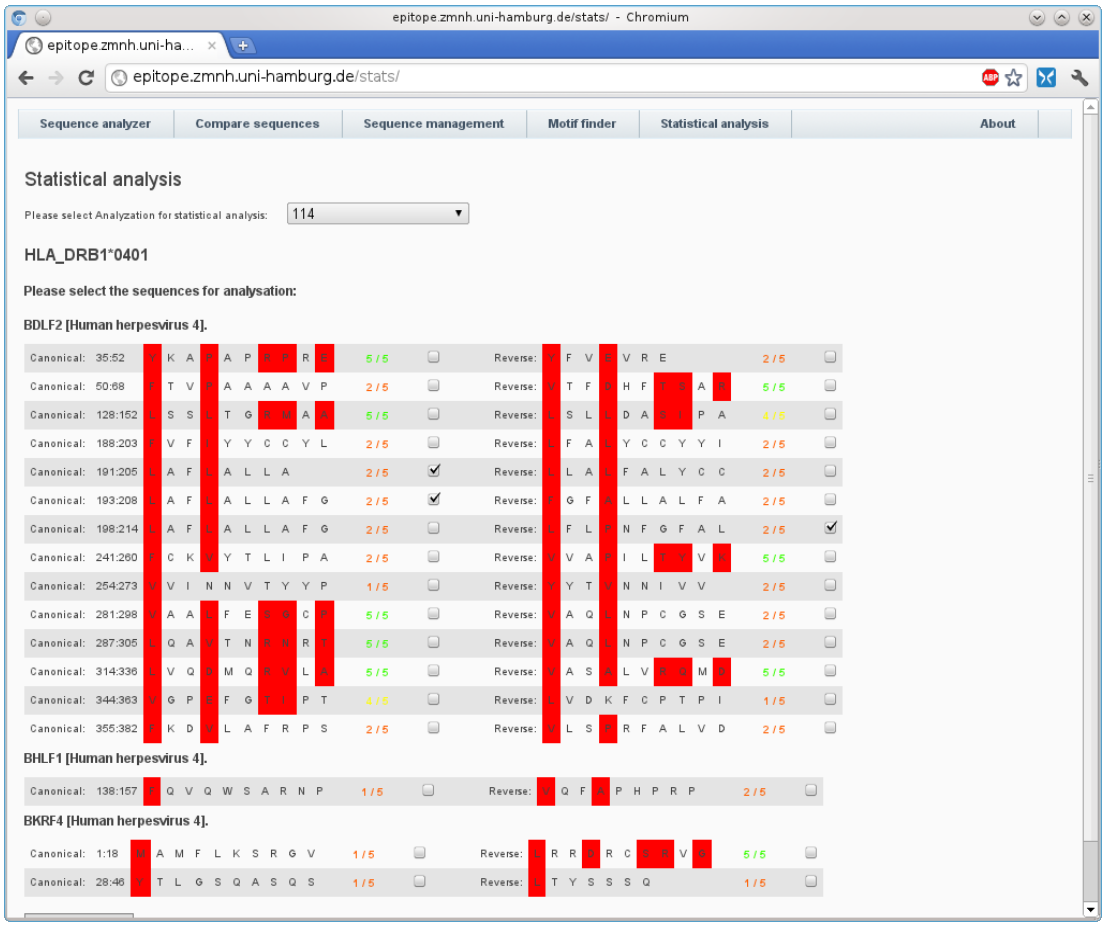


Abbildung 6.23: Statistik-Komponente

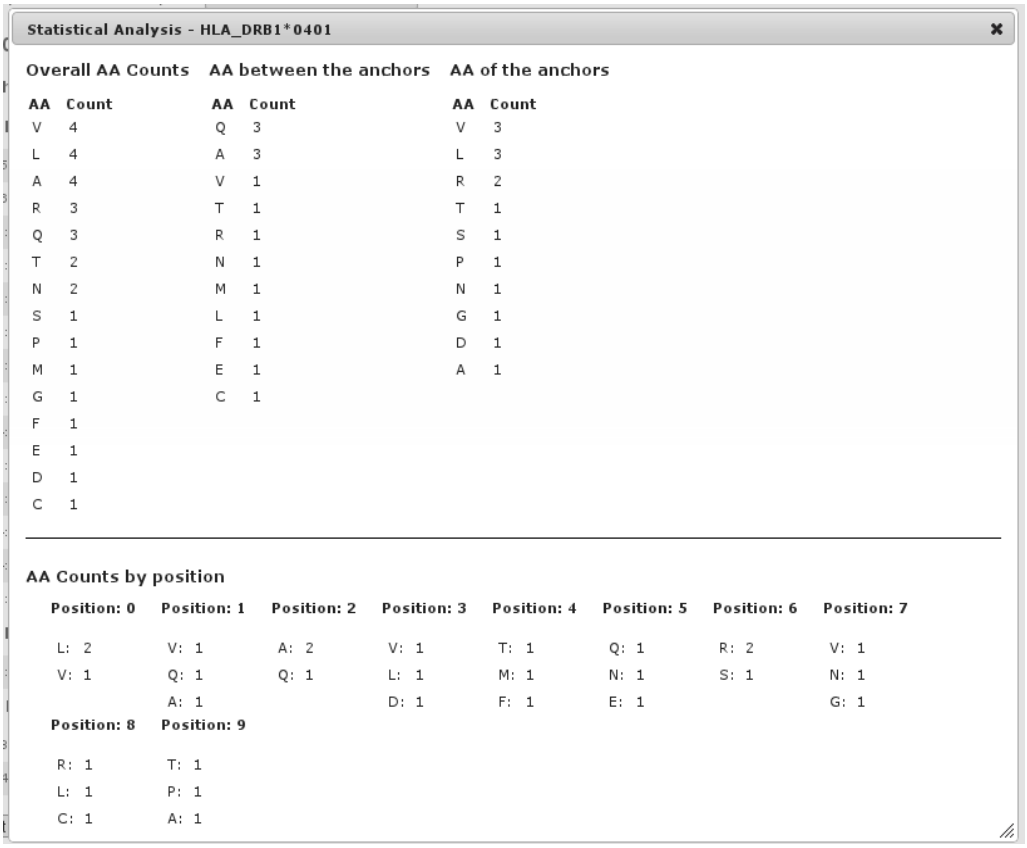


Abbildung 6.24: Ergebnisse der statistischen Analyse

6.8 Datenbankzugriff

Der Zugriff auf die Datenbank erfolgt über das durch Django implementierte objektrelationale Mapping (ORM). Es müssen keine SQL-Statements geschrieben werden, wodurch die Entwicklung vereinfacht und die Entwicklungszeit minimiert wird. Das Django ORM definiert die Tabellen der Datenbank als Klassen, ein Objekt der Klasse repräsentiert einen Datensatz.
Ein Datensatz wird durch die Instanziierung eines Objekts erzeugt. Die Attribute werden dabei mit Daten initialisiert. Der Datensatz wird über die Klassemethoden gespeichert, geändert und gelöscht.
Das Listing 6.3 zeigt die Klasse »Sequence«, die Stellvertreter für die Tabelle »Sequence« in der Datenbank ist.

```
1  class Sequence(Model):
2      url = CharField(max_length=200)
3      sequence = TextField()
4      datasheet = TextField()
5      last_fetch = DateTimeField(auto_now=True)
6      category = ForeignKey('SubCategory', blank=True)
7
8      def save():
9        # [...]
10       # Methode wurde von Oberklasse Model geerbt
11
12     def delete():
13       # [...]
14       # Methode wurde von Oberklasse Model geerbt
```

Listing 6.3: Beispiel für eine Klasse, die eine Datenbanktabelle repräsentiert

Die Klasse »Sequence« wird von der Klasse »Model« abgeleitet, die Bestandteil der Django-API für das objektrelationale Mapping ist. Sie stellt Methoden zum Aktualisieren, Anlegen und Löschen von Datensätzen in der Datenbank bereit. Das Löschen erfolgt über die geerbte Methode »delete()«, ein Datensatz wird angelegt oder aktualisiert, indem die ebenfalls geerbte Methode »save()« aufgerufen wird. Das Attribut »url« speichert eine Zeichenkette mit der maximalen Länge von 200 Zeichen, die Attribute »sequence« und »datasheet« sind Textfelder. Sie nehmen Freitexte beliebiger Länge[2] entgegen. Das Attribut »last_fetch« ist ein Datum- und Zeitfeld. Über den Parameter »auto_now = True« wird festgelegt, dass der Wert beim Anlegen und jeder Aktualisierung eines Datensatzes auf das aktuelle Datum und die aktuelle Uhrzeit gesetzt wird.
Das Attribut »category« ist ein Fremdschlüssel auf die Tabelle »SubCategory«. Durch die

[2] Eine Beschränkung findet auf Datenbankebene statt, wo eine maximale Größe für Datenbankfelder festgelegt wird.

Definition von Fremdschlüsseln wird eine Referenz auf andere Datenbankklassen erzeugt. Der Fremdschlüssel wird mit dem Parameter »blank = True« initialisiert, was bedeutet, dass beim Anlegen eines Datensatzes das Feld »category« frei bleiben kann.
Der in Listing 6.4 aufgeführte Programm-Quelltext zeigt, wie ein Datensatz der Tabelle »Sequence« hinzugefügt wird. In Zeile 1 wird ein Objekt der Klasse »Sequence« erzeugt, das als Stellvertreter für den Datensatz fungiert. Zeile 2, 3 und 4 füllen die Attribute mit Inhalt. Das Speichern erfolgt mit der Methode »save()« in Zeile 5. Die Django ORM-API erzeugt beim Aufruf der »save()«-Methode ein SQL-Insert-Statement, das den Datensatz in der Tabelle anlegt.

```
1 seq = Sequence()
2 seq.url = "http://www.ncbi.nlm.nih.gov/protein/YP_401695.1"
3 seq.datasheet = "DEFINITION  BDLF2 [Human herpesvirus 4]. [...]"
4 seq.sequence = "VDEQVAVEHGTVSHTISREEDGVVHERRVLASGERVEVFYKAPAPRPREGRASTFHDFTV"
5 seq.save()
```

Listing 6.4: Anlegen eines Sequence Datensatzes

Im Listing 6.5 wird ein vorhandener Datensatz editiert. Der Sequence-Datensatz aus der Datenbank wird anhand der Zeichenkette im Attribut »sequence« selektiert und geladen (Zeile 2). In Zeile 3 wird der Inhalt des Attributs verändert und Zeile 4 speichert den Datensatz in der Datenbank.

```
1 peptide = "VDEQVAVEHGTVSHTISREEDGVVHERRVLASGERVEVFYKAPAPRPREGRASTFHDFTV"
2 seq = Sequence.objects.get(sequence=peptide)
3 seq.sequence = "EEDGVVHERRVLA"
4 seq.save()
```

Listing 6.5: Beispiel für das Verändern eines Sequence Datensatzes

Die ORM-API von Django stellt wesentlich mehr Funktionalität bereit, als in den Beispielen verdeutlicht wurde. Es können komplexe Abfragen über mehrere Tabellen ausgeführt werden. Sofern eine Anfrage mehrere Datensätze zurückgibt wird die Datenstruktur »QuerySet« zurückgegeben, die das Setzen komplexer Filterregeln ermöglicht.

6.9 Softwaretest

Beim Testen von Software wird zwischen White-Box-Test und Black-Box-Tests unterschieden (vgl. Williams (2011)).
Beim White-Box-Test ist der Quelltext und die innere Struktur der Software bekannt. Es stehen verschiedene Testverfahren zur Auswahl: Unit-Test, Integrationstest und Regressionstest. Beim Unit-Test werden einzelne Klassen auf Funktionalität getestet. Beim Integrationstest wird getestet, ob die einzelnen Komponenten der Software miteinander kompatibel sind und ob ihr Zusammenspiel korrekt ist. Der Regressionstest wird durchgeführt, nachdem Änderungen vorgenommen wurden, somit wird geprüft, ob die Software weiterhin korrekt funktioniert. Im Regressionstest werden die Unit-Tests und die Integrationstests ausgeführt.
Im Black-Box-Test wird die Funktionalität der Software als Gesamtes getestet, ohne die innere Struktur und den Aufbau der Software zu kennen. Hierbei wird für bestimmte Eingaben geprüft, ob die durch die Software gelieferten Ausgaben korrekt sind.
Für die Software wurden im Rahmen dieser Arbeit sowohl White-Box-Tests als auch Black-Box-Tests durchgeführt.

White-Box-Tests

Während der gesamten Entwicklungsphase wurden mehrere Unit-Tests zum Testen der Softwaremodule erstellt, Django bietet hierfür eine spezielle API. Für jede »App« können Unit-Tests programmiert und über den Befehl »manage.py tests« gestartet werden. Die API zum Testen ist an das für Java entwickelte jUnit[3] angelegt. Dabei werden Klassen programmiert, deren Methoden die Tests implementieren. Ein Beispiel für einen Unit-Test mit Django ist im Listing 6.6 aufgeführt.
Die Methode »setUp()« wird vor den anderen Test-Methoden aufgerufen, dort können Klassenvariablen initialisiert werden, die für alle Test-Methoden verfügbar sein sollen.
Die mit »test_« im Methodennamen benannten Methoden implementieren die Tests und werden beim Starten nacheinander aufgerufen. Eine Prüfung über den Erfolg des Tests findet in der Methode »test_analysation« durch die Anweisung »self.assertEqual()« statt (im Listing 6.6 Zeile 8), hierbei wird der Inhalt von Variablen auf Inhaltsgleichheit geprüft. Sind die Inhalte verschieden, wird die Test-Methode abgebrochen und der Benutzer darüber informiert.

[3] http://www.junit.org

```
1  class AnalyserTest(TestCase):
2      def setUp(self):
3          [...]
4
5       def test_analysation(self):
6          [...]
7          res = do_analyse("AFFGE")
8          self.assertEqual(5.75, res)
9
10      def test_db_write(self):
11         [...]
```

Listing 6.6: Django Unit-Test

Es wurden verschiedene Unit-Tests zum Prüfen der einzelnen Klassen implementiert. Unter anderem wird getestet, ob die Schnittstelle zu dem »MHC-II binding prediction«-Programm richtige Werte für die Bindungswahrscheinlichkeit liefert oder die SYFPEITHI-Anker-Motive korrekt gefunden werden.
Neben dem Testen auf Klassenebene ist eine weitere Möglichkeit die Webanwendung zu testen das Simulieren von Browseraufrufen und Erzeugen von Anfragen.
Zum Erzeugen von Http-Requests bietet die Django Testing-API einen Http-Client, der von den Testklassen verwendet wird. Hiermit können Aufrufe an die Webanwendung simuliert und das zurückgegebene Ergebnis geprüft werden. Beim Starten der Tests wird dafür ein minimaler Webserver gestartet. Die Testklassen haben sowohl direkten Zugriff auf die Datenbank als auch die Möglichkeit, Http-Requests an den Testserver zu senden. Somit wird es möglich einen Http-Request zu erzeugen, der eine View anweist, einen Datensatz in der Datenbank zu speichern. Durch den Zugriff auf die Datenbank kann verifiziert werden, ob der Datensatz korrekt erstellt wurde.
Dieses Verfahren wurde verwendet, um die einzelnen Views automatisiert zu testen. Jede View wird durch den Testclient aufgerufen und geprüft, ob ein Ergebnis geliefert oder ein Fehler erzeugt wurde.

Black-Box-Tests

Black-Box-Tests wurden nach jedem größeren Entwicklungsschritt durch die Wissenschaftler des ZMNH und den Entwickler selbst durchgeführt.
Für die Verwaltung der Fehlerberichte wurde die Software »Trac[4]« verwendet. Trac bietet ein integriertes Wiki für die Dokumentation und stellt den Benutzern eine Möglichkeit bereit, Fehlerberichte (»Tickets«) zu erstellen. Diese können kategorisiert und priorisiert werden.

[4] http://trac.edgewall.org

Die Berechnung der Bindungswahrscheinlichkeit wurde verifiziert, indem eine Analyse parallel mit der entwickelten Software und der »IEDB«-Weboberfläche für das »MHC-II binding prediction«-Programm durchgeführt und die errechneten Bindungswahrscheinlichkeiten anschließend verglichen wurden.
Durch die Analyse einer hohen Proteinanzahl für verschiedene MHC-Moleküle wurde ein Lasttest durchgeführt. Während der Berechnung wurde geprüft, ob die Software bedienbar bleibt und wie sich die Anfragen auf die serverseitige Last auswirken.
Die Weboberfläche wurde in den Browsern »Google Chrome«, »Safari«, »Firefox«, »Opera« und »Internet Explorer« unter verschiedenen Betriebsystemen getestet.

7 Schluss

7.1 Zusammenfassung

Im Rahmen der Bachelorarbeit wurde eine Software entwickelt, die es erstmalig ermöglicht, eine große Menge an Proteinsequenzen auf ihre Bindungseigenschaften für verschiedene MHC-Moleküle zu untersuchen und die Ergebnisse auszuwerten.
Als Basis für die Analyse wird wissenschaftlichen Proteindatenbank NCBI verwendet und die dort gespeicherten Proteinsequenzen in eine eigene Datenbank importiert. Eine Verwaltungs-Komponente ermöglicht es, die gespeicherten Proteinsequenzen einzusehen und für die Analysen zu selektieren.
Die Proteinsequenzen werden mit zwei Programmen der Bioinformatik analysiert: dem »MHC-II binding prediction«-Programm und SYFPEITHI. Das »MHC-II binding prediction«-Programm wurde in die Software integriert, indem eine Schnittstelle programmiert wurde. SYFPEITHI wurde durch die Implementierung einer Komponente zum Finden der Anker-Motive integriert. Die Anker-Motive können durch den Benutzer angelegt und in der Datenbank gespeichert werden.
Die durch das »MHC-II binding prediction«-Programm gewonnenen Analyseergebnisse werden durch Grafiken und tabellarische Ausgaben dem Benutzer veranschaulicht und können in das MS-Excel-Format exportiert werden.
Für die Analyse stehen verschiedene Komponenten zur Verfügung: Der »Sequence Analyser«, »Sequence Comparer« und »Motif Finder«. Der »Sequence Analyser« kann die Proteinsequenz importieren und mit dem »MHC-II binding prediction«-Programm die Bindungswahrscheinlichkeiten bestimmen. Der Benutzer hat die Möglichkeit, aus verschiedenen Visualisierungs-Techniken zu wählen. Mit dem »Sequence Comparer« werden mehrere Proteinsequenzen parallel analysiert. Der »Motif Finder« bestimmt die Bindungswahrscheinlichkeit beliebig vieler Proteinsequenzen und findet die Anker-Motive in den Peptiden, deren Bindungswahrscheinlichkeit einen festgelegten Schwellenwert unterschreiten. Die Ergebnisse der Motivsuche werden ebenfalls in der Datenbank gespeichert und können später mit einer Statistik-Komponente untersucht werden.
Die Visualisierung und Statistik-Komponente ermöglichen die systematische Auswertung großer Datenmengen und erlauben es, Zusammenhänge im Bindungsverhalten von Peptidketten zu erkennen. Dadurch wird es möglich, das Verhalten der T-Zell-Rezeptoren vorherzusagen.

Implementiert wurde die Software in der Programmiersprache Python mit dem Webframework Django. Bei der Software handelt es sich um eine »Rich Internet Application«, deren Weboberfläche an eine Desktop Anwendung angelegt ist. Für die Entwicklung der Weboberfläche wurde auf die JavaScript Bibliothek jQuery und das jQuery-Plugin jQueryUI zurückgegriffen.

7.2 Fazit

Bereits während der Entwicklung haben die Wissenschaftler des ZMNH begonnen, mit der Software zu arbeiten und Verbesserungsvorschläge, Änderungswünsche und Fehlermeldungen abgegeben. Die grafischen und tabellarischen Darstellungen wurden in Publikationen übernommen.
Bei Abschluss der Bachelorarbeit waren bereits umfangreiche Analysen von 80 Proteinen des »Eppstein-Barr-Virus« im System enthalten. Das »Eppstein-Barr-Virus« steht im Verdacht, die Multiple Sklerose Erkrankung wesentlich zu beeinflussen.
Damit die Wissenschaftler standortunabhängig auf die Software zugreifen können, wurde sie unter der Adresse http://epitope.zmnh.uni-hamburg.de in das Internet gestellt. Damit wird einem größeren Kreis von Wissenschaftlern das Arbeiten mit der Software ermöglicht.
Eine wissenschaftliche Publikation der Software und der damit erzielten Ergebnisse ist in Vorbereitung.

7.3 Mögliche Verbesserungen

Die Usability der Software wurde bisher nicht untersucht und ist an ein paar Stellen eventuell verbesserungswürdig. Es fehlt die Integration einer Hilfefunktion. Hilfetexte, die einerseits die Funktionalität der Benutzeroberflächen beschreiben und andererseits Hilfestellung bei der Auswahl der richtigen Parameter geben, wären wünschenswert.
Die Kompatibilität mit mobilen Endgeräten, wie z.B. Android-Smartphones oder dem iPhone/iPad, wurde nicht getestet. Da es sich um eine Webanwendung handelt, ist es durchaus denkbar, dass die Software mit Smartphones oder Tablet-PCs benutzt werden wird. Die Weboberfläche wurde bisher nicht auf die Verwendung mit Touch-Eingabegeräten optimiert.
Bisher ist in der Software keine Benutzerauthentifizierung vorgesehen, alle Benutzer arbeiten auf dem gleichen Datenbestand. Jeder kann Daten löschen oder einsehen. Sobald die Software für einen breiten Benutzerkreis geöffnet wird, müssen eine Authentifizierung und Benutzerprofile implementiert werden. Hierdurch kann jeder Anwender einen individuellen Datenbestand vorhalten, auf den nur er vollen Zugriff hat. Denkbar wäre auch, dass die Anwender Daten für andere Benutzer freigeben können.

7.4 Ausblick

Die Software wird in Zukunft weiterentwickelt werden.
Die statistische Auswertungen lassen sich nahezu beliebig erweitern: In der aktuellen Implementierung werden die Häufigkeiten der Aminosäuren an verschiedenen Positionen innerhalb des Peptids dem Benutzer präsentiert. Denkbar wäre es, mit Techniken der künstlichen Intelligenz die Peptide zu analysieren, um Muster oder Zusammenhänge in den Daten zu erkennen, anhand derer auf die Bindungseigenschaften geschlossen werden kann.
Weiterhin besteht der Wunsch, ein »Basic Local Alignment Search« (BLAST) in die Anwendung zu integrieren. Über dieses Verfahren werden Peptid- oder Proteinsequenzen auf Ähnlichkeiten untersucht. Damit könnte automatisch in ausgewählten Peptidketten mit guten Bindungseigenschaften und geeigneten Anker-Motiven nach strukturähnlichen Proteinen gesucht werden. Diese Funktion konnte im Rahmen der Bachelorarbeit nicht mehr implementiert werden.

Literaturverzeichnis

[ncbi 2011] : *National Center for Biotechnology Information, des National Institute of Health*. 2011. – URL http://www.ncbi.nlm.nih.gov/

[Alchin 2010a] ALCHIN, Marty: *Pro Django*. Apress, 2010. – ISBN 978-1430210474

[Alchin 2010b] ALCHIN, Marty: *Pro Python*. Apress, 2010. – ISBN 978-1430227571

[Bennett 2008] BENNETT, James: *Practical Django Projects*. Apress, 2008. – ISBN 978-1590599969

[Bibeault und Katz 2010] BIBEAULT, Bear ; KATZ, Yehuda: *jQuery in Action*. Manning, 2010. – ISBN 978-1935182320

[Breu u. a. 2005] BREU, R. ; MATZNER, Th. ; NICKL, F. ; WIEGERT, O.: *Software Engineering*. Oldenburg, 2005. – ISBN 3-486-57574-0

[Chen 1976] CHEN, Peter Pin-Shan: The Entity-Relationship Model – Toward a Unified View of Data. In: *ACM Transaction on Database Systems* 1 No. 1 (1976), March, S. 9–36

[Eby 2011] EBY, P.J.: *Python Enhancement Proposal für das »Python Web Server Gateway Interface«*. 2011. – URL http://www.python.org/dev/peps/pep-3333/

[Felicitas Witte 2011] FELICITAS WITTE, Dr. m.: *Netdoktor - Artikel zu Multiple Sklerose - Ursachen, Symptome, Diagnose*. 2011. – URL http://www.netdoktor.de/Krankheiten/Multiple-Sklerose/

[Fischer 2005] FISCHER, Gerd: *Stochastik einmal anders: Parallel geschrieben mit Beispielen und Fakten, vertieft durch Erläuterungen*. Vieweg+Teubner, 2005. – ISBN 978-3528039677

[Foundation 2011] FOUNDATION, Django S.: *Django – frequently asked questions*. 2011. – URL http://docs.djangoproject.com/en/1.3/faq/general/

[Hogan 2011] HOGAN, Brian P.: *HTML5 and CSS3: Develop with Tomorrow's Standards Today*. Pragmatic Programmers, 2011. – ISBN 978-1934356685

[Holovaty 2010] HOLOVATY, Adrian: *The Definitive Guide to Django: Web Development Done Right*. Apress, 2010. – ISBN 978-1430219361

[Janeway u. a. 2009] JANEWAY, Charles A. ; MURPHY, Kenneth M. ; TRAVER, Paul ; WALPORT, Mark: *Janeway Immunologie*. Spektrum Akademischer Verlag, 2009. – ISBN 3-8274-1079-7

[Jürgen Groth 2011] JÜRGEN GROTH, Dr. rer. n.: *Meine Moleküle. Deine Moleküle - Online Buch*. 2011. – URL `http://www.meine-molekuele.de/`

[Kim u. a. 2009] KIM, Yohan ; SIDNEY, John ; PINILLA, Clemencia ; SETTE, Alessandro ; PETERS, Bjoern: Derivation of an amino acid similarity matrix for peptide:MHC binding and its application as a Bayesian prior. In: *BMC Bioinformatics* 10:394 (2009), November

[Lesk 2003] LESK, Arthur M.: *Bioinformatik - Eine Einführung*. Spektrum, Akad. Verlag, 2003. – ISBN 3-8274-1371-0

[Lutz 2011] LUTZ, Mark: *Programming Python*. O'Reilly Media, 2011. – ISBN 978-0596158101

[Meyer 2007] MEYER, Eric A.: *CSS. Das umfassende Handbuch*. O'Reilly, 2007. – ISBN 978-3897214934

[Neumann 2008] NEUMANN, Jürgen: *Immunbiologie*. Springer Berlin Heidelberg, 2008 (Springer-Lehrbuch). – ISBN 978-3-540-72569-5

[Oestereich 2006] OESTEREICH, Bernd: *Analyse und Deisgn mit UML 2.1*. Oldenbourg, 2006. – ISBN 3-486-57926-6

[Rammensee u. a. 1999] RAMMENSEE, Hans-Georg ; BACHMANN, Jutta ; EMMERICH, Niels Philipp N. ; BACHOR, Oskar A. ; STEVANOVIĆ, Stefan: SYFPEITHI: database for MHC ligands and peptide motifs. In: *Immunogenetics* (1999), Nr. 50, S. 213–219

[Reenskaug 1979] REENSKAUG, Trygve: Thing-Model-View-Editor – An example from a planningsystem. (1979), May

[Silberschatz u. a. 2010] SILBERSCHATZ, Abraham ; KORTH, Henry F. ; SUDARSHAN, S.: *Database System Concepts*. Mcgraw-Hill Professional, 2010. – ISBN 0071289593

[Wang u. a. 2008] WANG, Peng ; SIDNEY, John ; DOW, Courtney ; MOTHÉ, Bianca ; SETTE, Alessandro ; PETERS, Bjoern: A Systematic Assessment of MHC Class II Peptide Binding Predictions and Evaluation of a Consensus Approach. In: *PLoS Computational Biology* volume 4 issue 4 (2008), April

[Williams 2011] WILLIAMS, Prof. L.: *Software Engineering Abschnitt auf der OpenSeminar Homepage*. 2011. – URL `http://openseminar.org/se/screen.do`

Anhang

A Screenshots der Weboberfläche

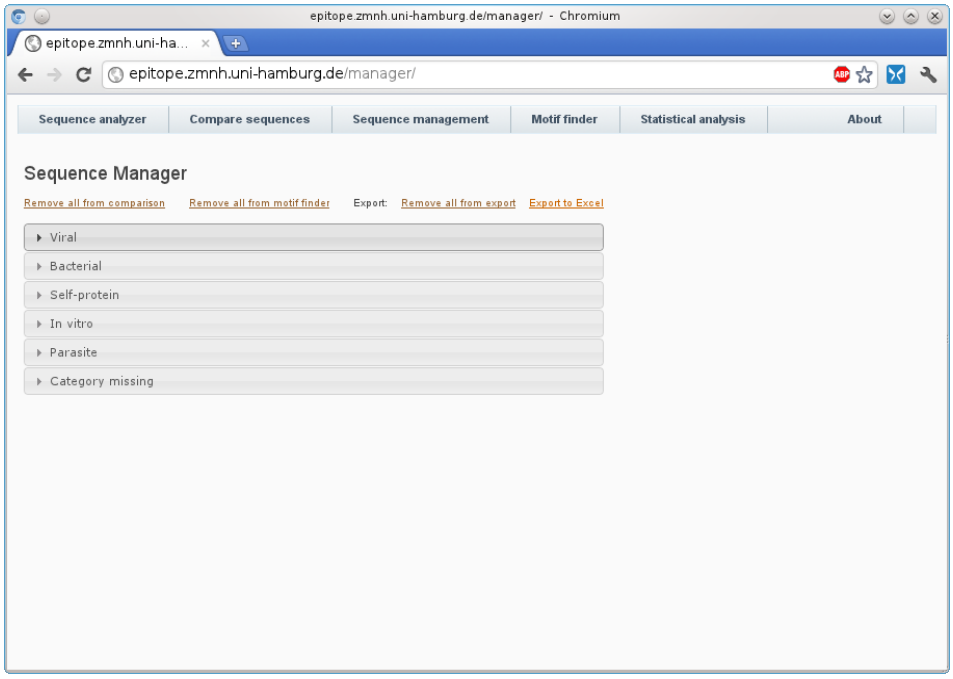


Abbildung A.1: »Sequence Manager«-Komponente mit geschlossenen Kategorien

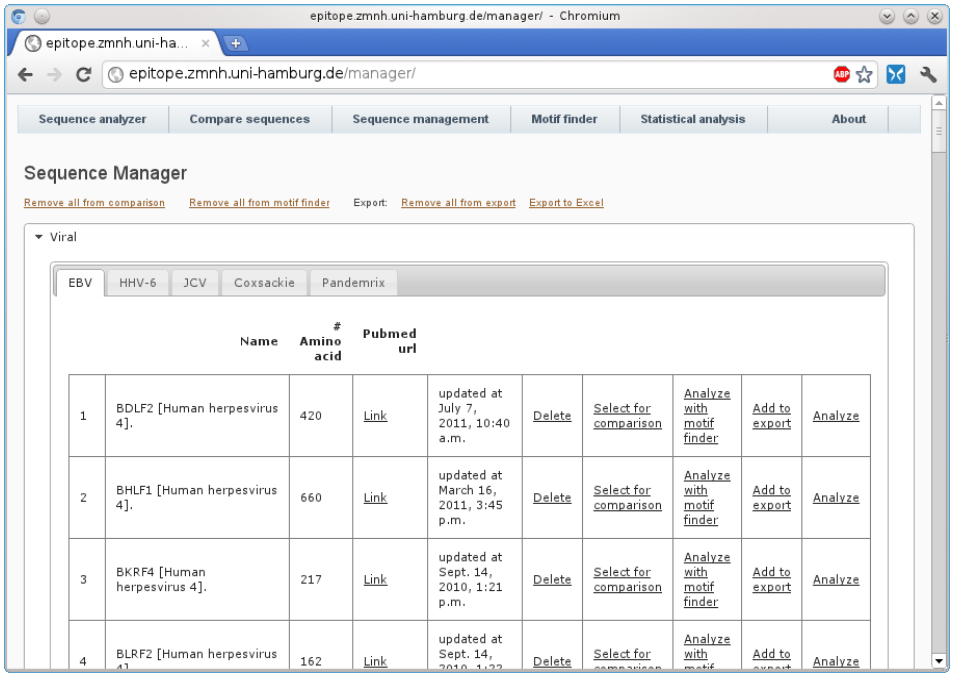


Abbildung A.2: »Sequence Manager«-Komponente mit geöffneter Kategorie

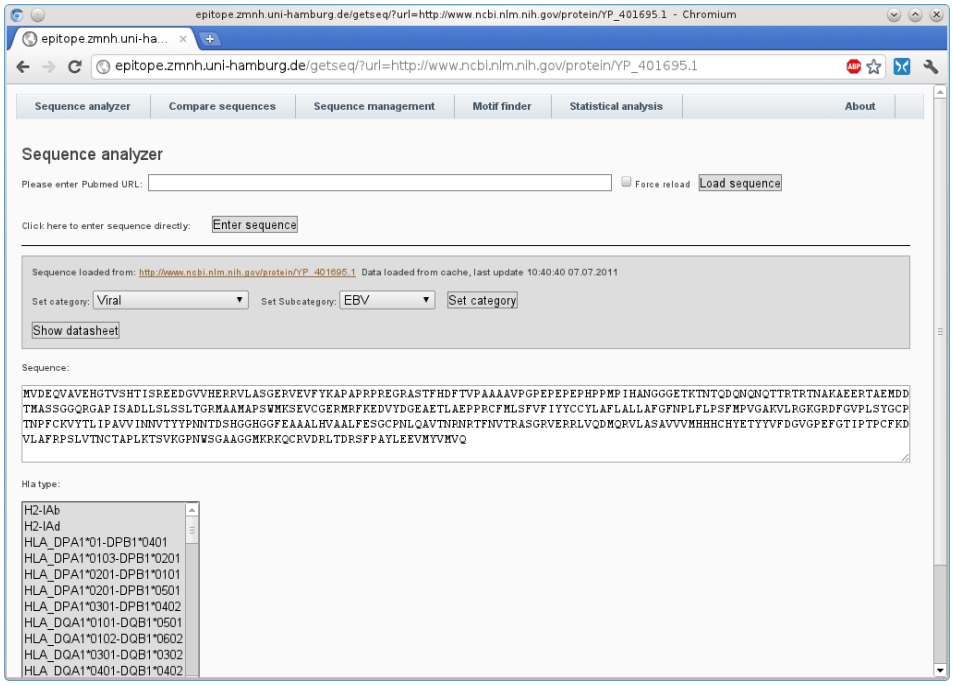


Abbildung A.3: Bedienoberfläche der »Sequence Analyser«-Komponente

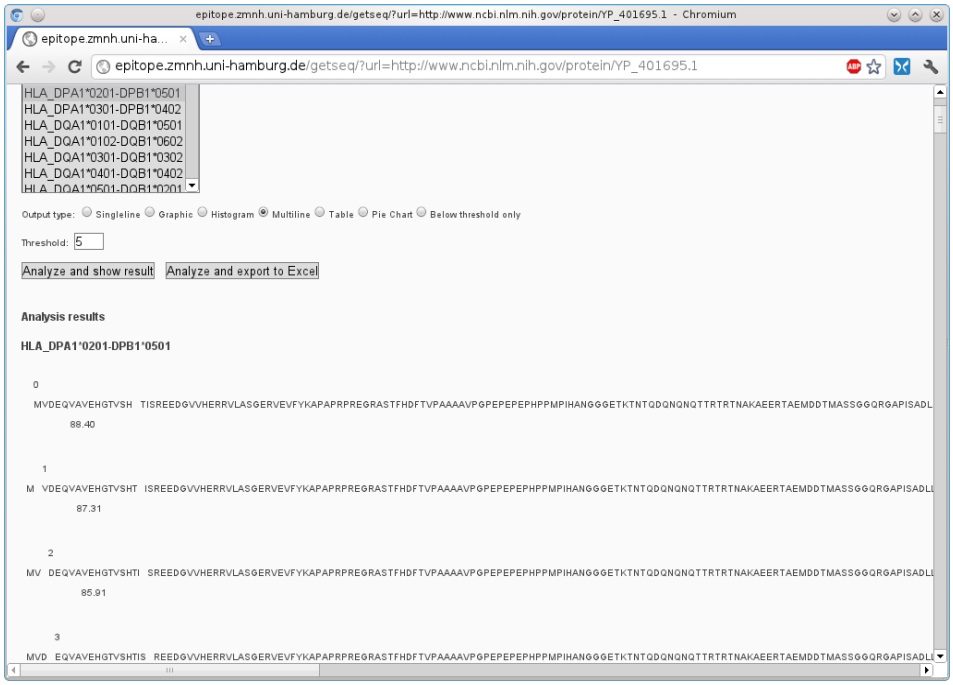


Abbildung A.4: Anzeige des Analyseergebnis in der »Sequence Analyser«-Komponente

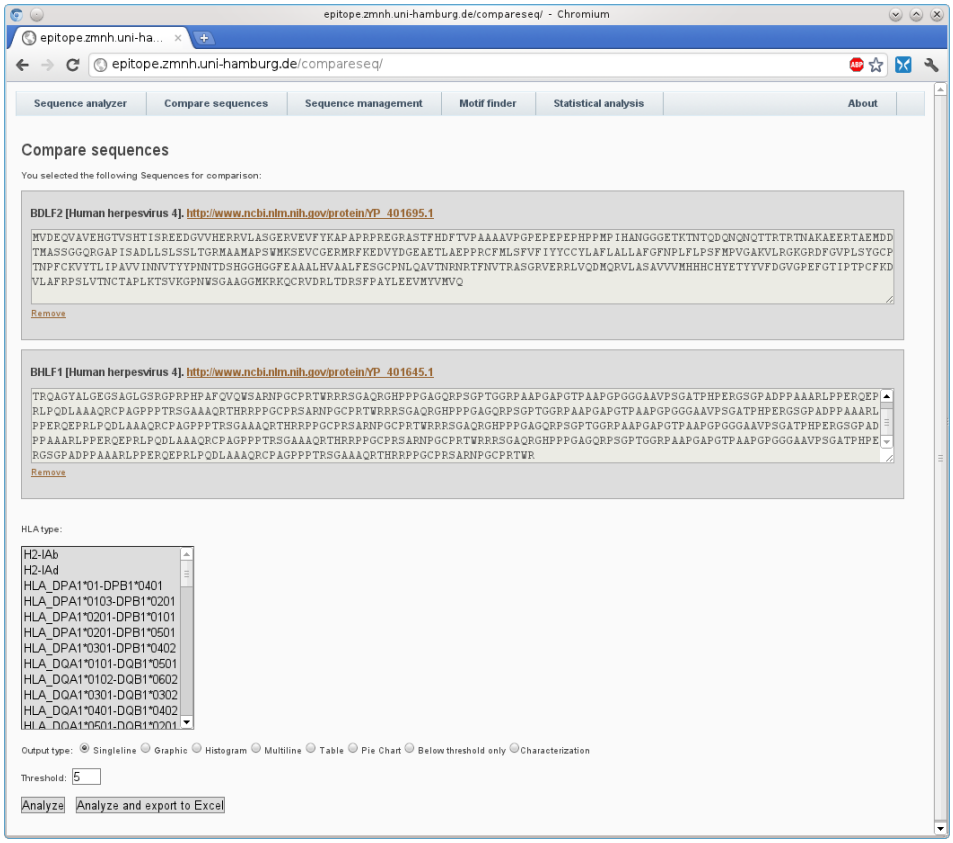


Abbildung A.5: »Sequence Comparer«-Komponente

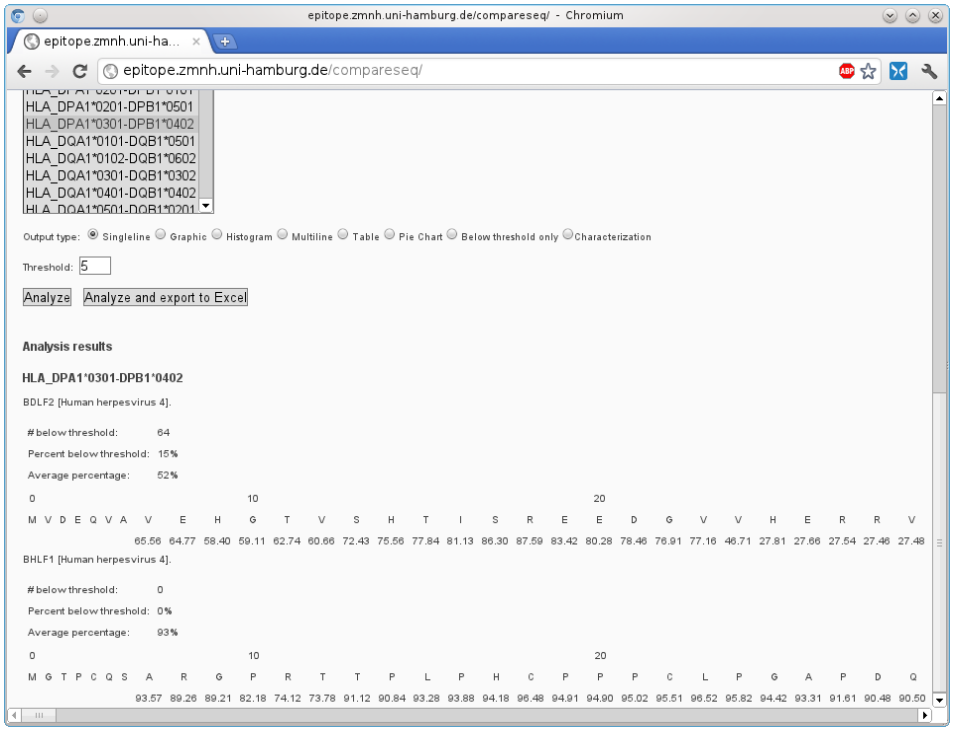


Abbildung A.6: Anzeige des Analyseergebnis in der »Sequence Comparer«-Komponente

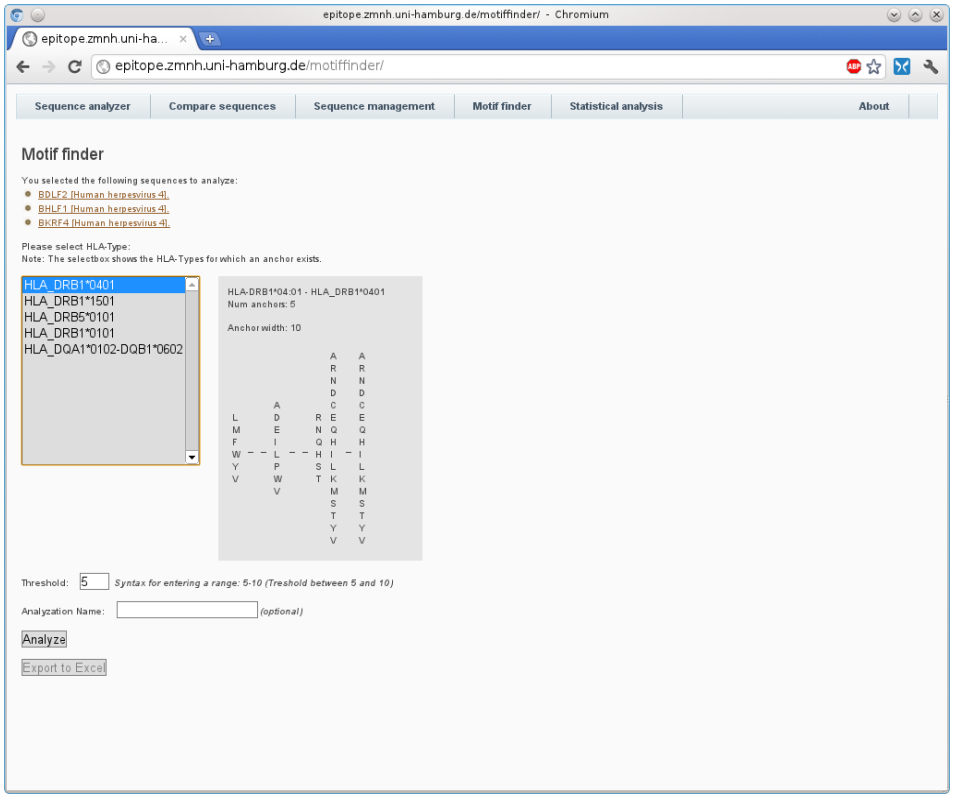


Abbildung A.7: Auswahl eines Ankers in der »Motif-Finder«-Komponente

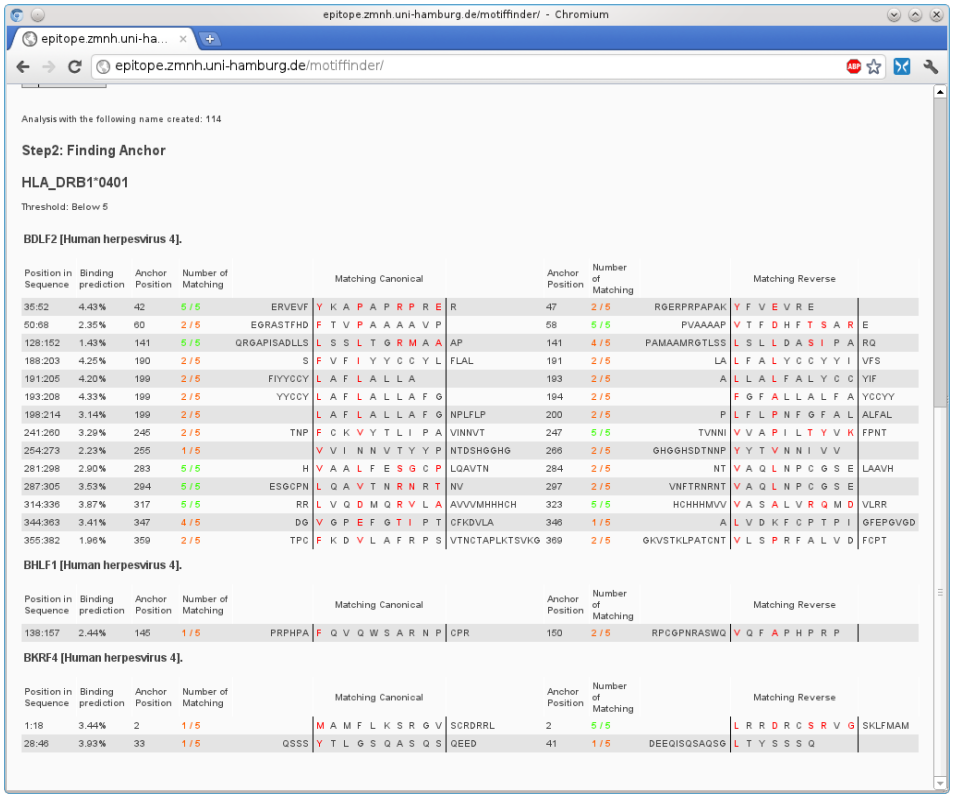


Abbildung A.8: »Motif-Finder«-Komponente

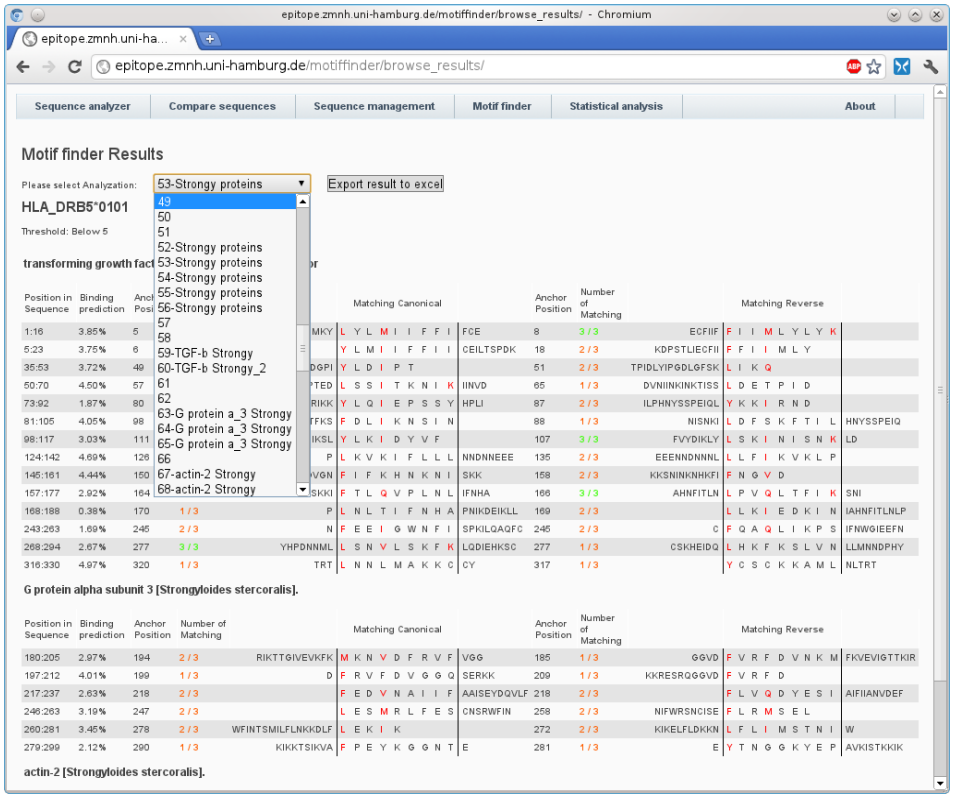


Abbildung A.9: Anzeigen der Analyseergebnisse der »Motif-Finder«-Komponente

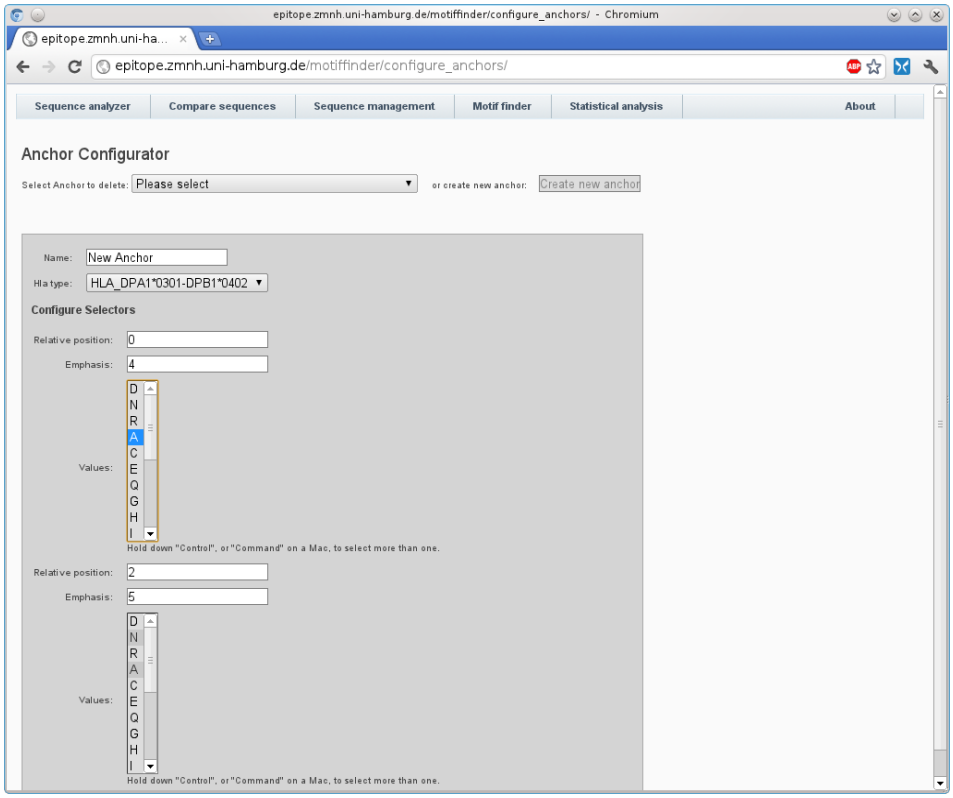


Abbildung A.10: Konfiguration der »Motif-Finder«-Komponente

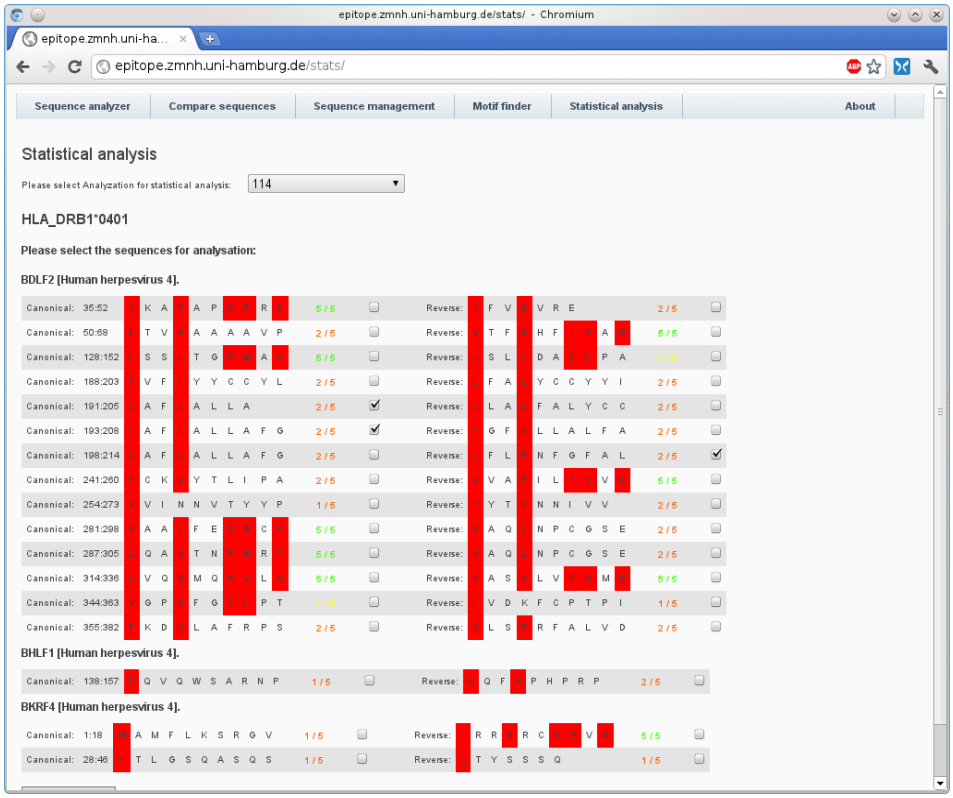


Abbildung A.11: Auswahl von Peptiden für die Statistik

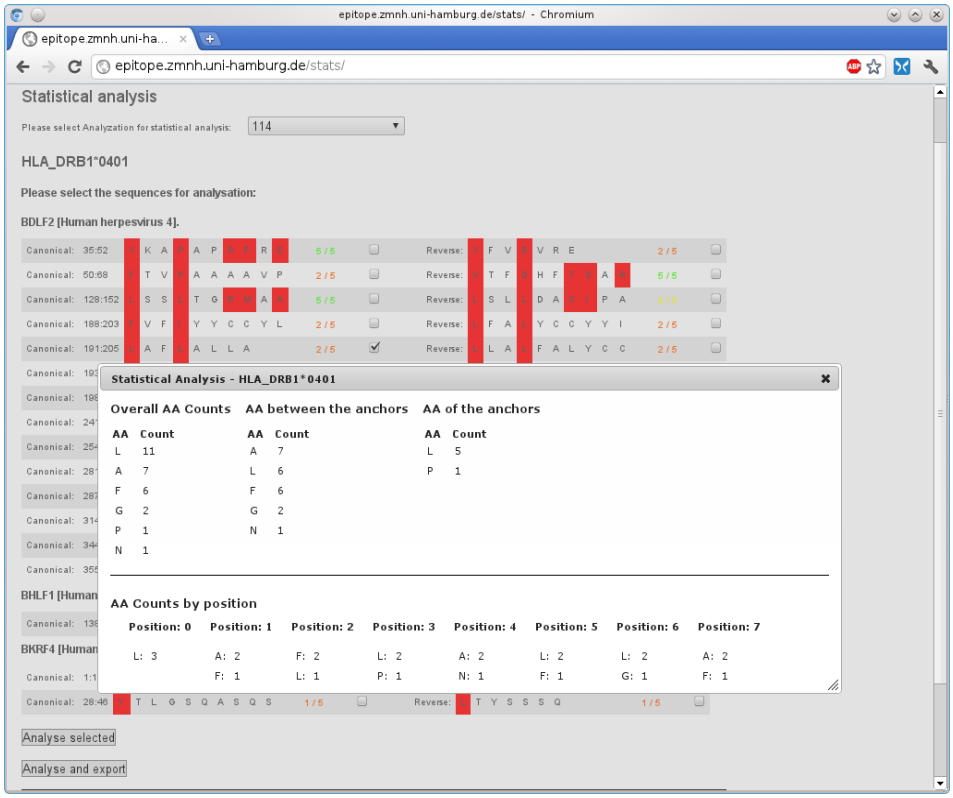


Abbildung A.12: Anzeige der Ergebnisse der statistischen Analyse

B Codebeispiele

Oberflächen mit jQueryUI

Die folgenden beiden Beispiele zeigen die Programmierung einer grafischen Oberfläche mit jQueryUI.

```
1  <html>
2  <head>
3  <script type="text/javascript" src="/jquery.min.js"></script>
4  <script type="text/javascript" src="/jquery ui.min.js"></script>
5  <script type="text/javascript">
6  $(document).ready(function() {
7     $('input[name=open_dialog]').click(function() {
8        $('#show_as_dialog').load('/dialog_content.html', function() {
9           $(this).dialog();
10       }
11    });
12 });
13 </script>
14 </head>
15 <body>
16 <div id="#show_as_dialog">
17 </div>
18 <input type="button" name="open_dialog" />
19 </body>
20 </html>
```

Listing B.1: Laden und Anzeigen eines Dialogfenster mit jQueryUI

```
1  <html>
2  <head>
3  <script type="text/javascript" src="/jquery.min.js"></script>
4  <script type="text/javascript" src="/jquery ui.min.js"></script>
5  <script type="text/javascript">
6  $(document).ready(function() {
7     $('#accordion').accordion();
8  });
9  </script>
10 </head>
11 <body>
12 <div id="accordion">
13    <h3><a href="#">Viral</a></h3>
14    <div>[...] Inhalt [...]</div>
15    </div>
16    <h3><a href="#">Bacterial</a></h3>
17    <div>[...] Inhalt [...]</div>
18 </body>
19 </html>
```

Listing B.2: Erzeugen eines »Accordion«-Widget mit jQueryUI

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 25. August 2011	
Ort, Datum	Unterschrift