**Diplomarbeit**

# Programmieren einer Datenbank zur Verwaltung gentechnischer Konstrukte


eingereicht von

**Thomas Strasser**


zur Erlangung des akademischen Grades

**Doktor der gesamten Heilkunde**

**(Dr. med. univ.)**


an der

**Medizinischen Universität Graz**

ausgeführt am

**Institut für Biophysik**

unter der Anleitung von

**Ao.Univ.-Prof. Dr.phil. Schreibmayer Wolfgang**

Zweitbetreuer:

**Ass.-Prof. Dipl.-Ing. Dr.techn. Platzer Dieter**



Graz, Mai 2016

*Eidesstattliche Erklärung*

*Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst habe, andere als die angegebenen Quellen nicht verwendet habe und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.*

*Graz, am 24. Mai 2016* *Thomas Strasser eh*

# Danksagungen

Zuallererst möchte ich natürlich **meinen Eltern** danken, die mir dieses Studium ermöglicht haben. Ohne sie wäre dies wohl nicht möglich gewesen.

Auch ein besonderer Dank gilt meinen Klassenvorständen in der Hauptschule (**Karl Wöllinger**) und in der HTL Grieskirchen (**DI. Wolfgang Kaiser**). Durch Herrn DI Wolfgang Kaiser durfte ich die mehr oder weniger hohe Kunst der Softwareentwicklung erlernen, was für diese Art von Diplomarbeit wohl unerlässlich ist.

Auch gilt mein Dank dem **Österreichischen Roten Kreuz**, welches mich durch den Zivildienst und bei meiner späteren ehrenamtlichen Tätigkeit auf meine ‚soziale' Ader hingewiesen hat. Ohne dieses Engagement hätte ich wohl nie an ein Studium der Humanmedizin gedacht.

Auch möchte ich meinen beiden Diplomarbeits-Betreuern Herrn **Ao.Univ.-Prof. Dr.phil. Schreibmayer Wolfgang** und Herrn **Ass.-Prof. Dipl.-Ing. Dr.techn. Platzer Dieter** danken, die mir immer mit Rat und Tat zur Seite gestanden sind. Auch wenn ich mir mit dieser Diplomarbeit oftmals sehr viel Zeit gelassen habe – ich konnte mir ihrer Unterstützung sicher sein.

Nicht zu vergessen ist auch Frau **Gorischek Astrid**, die mir von der Anwenderseite her Tipps gab, wie die Oberfläche am besten auszusehen haben soll.

Zu guter Letzt möchte ich mich bei den Studentenverbindungen, denen ich angehöre, bedanken. Diese wären: **K.Ö.St.V. Rugia Ried, K.St.V. Stubenberg Bruck, K.Ö.St.V. Traungau Graz**. Diese drei Vereine haben mir gezeigt, dass es neben dem Studium auch noch viele andere interessante Sachen zu erleben gibt. Auch konnte ich hier meine Soft-Skills trainieren – welche ein wichtiger Eckpfeiler für mein späteres Leben sein werden.

Nun wirklich das letzte Wort gilt all **meinen Freunden** die mich immer wieder mit aufmunternden Worten und Gesprächen dahin gebracht haben wo ich heute bin. **Danke!!!**

# Zusammenfassung

Einleitung: Die Gentechnik ist aus der heutigen Medizin nicht mehr wegzudenken. Um Information zu den Bausteinen für gentechnische Konstrukte und der Konstrukte selbst effizient speichern zu können, wurden diese Daten von Mitarbeitern des Instituts für Biophysik bisher in Form von Excel-Dateiblättern, verschiedenen physischen Ordnern (Ausdrucke!) und in diversen anderen Fileformaten auf verschiedenen Speicherorten hinterlegt. Die Information zu den Konstrukten untergliedert sich in Information zu verschiedenen Restriktionsenzymen, Primersequenzen, Vektoren, Gene sowie Gen-Produkten.
Da für die Informationsspeicherung kein geeignetes Softwareprodukt existierte, wurde ein solches programmiert. Dieses soll auf die Anforderungen der Forschungs-Mitarbeiter zugeschnitten und einfach aktualisierbar sein.
Material und Methoden: Neben einer Software-Entwicklungsumgebung, wie dem Microsoft Visual Studio 2013 Professional, wurde für die Entwicklung der Software ein relationales Datenbanksystem benötigt. Hier bot sich der MSSQL Server 2014, ebenfalls von Microsoft, an. Um etwaige Dokumentationen für die Konstrukte nicht in der Datenbank ablegen zu müssen – hier haben manche Pendants nur eine beschränkte Speicherkapazität – wurden größere Dateien auf einen FTP Server abgelegt.
Ergebnisse: Die Oberfläche des Clients wurde mithilfe von Tabs realisiert. Diese haben jeweils einen einheitlichen Aufbau – oben die Eingabemaske und unten werden die bereits bestehenden Datensätze angezeigt. Diese Daten werden in insgesamt neunzehn Tabellen am SQL Server abgebildet und unter weiterer Zuhilfenahme von neun Views ausgelesen. Änderungen in den Daten werden mit einer Push-Benachrichtigung jedem Client mitgeteilt.
Diskussion: Am Ende eines agilen Entwicklungsprozesses wurde eine Client-Software für die Verwaltung der gentechnischen Konstrukte realisiert.
Der Client kann inklusive Quellcode dem Institut übergeben werden. Es zeigten sich weitere Möglichkeiten wie eine Umsetzung mithilfe der Programmiersprache ASP.NET oder Java, welche aber neben Vor- auch Nachteile haben.
Auch wurde eine Erweiterung der Software mithilfe von Programmbibliotheken (.DLLs) diskutiert.

## Abstract

Introduction: Genetic engineering is very important in the contemporary medical sector. The department of biophysics and its members carefully processed data in Microsoft Excel, and made use of hardcopies in selected files and different other formats on varying memory locations, in order to collect and efficiently save/store data of genetically engineered constructs. The information dealing with these constructs can be differentiated in data on different restriction enzymes, primer sequences, vectors, genes as well as genetically-modified products.

Up to the present, no such software was available, which could save data and other information. Consequently, a software has been developed. This software should be tailored to meet the requirements of researchers and people alike. Further, it should be an easily updatable software.

Material and Methods: Besides a software-development-program, such as Microsoft Visual Studio 2013 Professional, a relational database system was needed. At this point, MSSQL Server 2014 was used, which is also a program by Microsoft. To prevent further documentation of the constructs on a certain database – some pendants only have restricted storage capacity – bigger files were stored on a FTP Server.

Results: The surface area of the client was developed by means of tabs. These tabs have a standardized structure. On top, there is an input mask and below one can find the already existing datasets. This data is present in additional nineteen tables on the SQL Server and is processed with addtional nine views. Data modification is realized by so called push-messages, which are sent to each individual client.

Discussion: A client-based software has been developed at the end of a agile development process in order to administer genetically-engineered constructs. The client (including its source code) was transfered tot he department of biophysics. During the development process, further possibilities concerning the software were discovered, such as the implementation by means of the programming language ASP.NET or Java, which have both advantages and disadvantages. Further, an extension of the software by the use of program libraries (.DLLs) has been discussed.

# Inhaltsverzeichnis

# Glossar und Abkürzungen

| | |
|---|---|
| SQL | Structured Query Language |
| FTP | File Transfer Protocol |
| SVN | Apache Subversion |
| DNA | desoxy ribonucleic acid |
| mRNA | messenger ribonucleic acid |
| ASP | active server pages |
| GUID | Globally Unique Identifier |
| MCS | Multiple cloning site |
| CPOL | Code Project Open License |

# Abbildungsverzeichnis

# 1. Einleitung

Die Verwaltung von gentechnischen Konstrukten ist für Institute wie die Biophysik eine wichtige Komponente ihrer täglichen Forschungsarbeit. Neben der Entwicklung und Zusammenstellung der Konstrukte ist auch eine effektive Lagerhaltung dieser gentechnischen Konstrukte von großer Bedeutung.

Diese Lagerhaltung geschieht mittels sogenannter ‚Batches'. Dabei handelt es sich um die Summer der Aliqoute eines Produktionsprozesses. Jeder Batch wird individuell überprüft und charakterisiert, da die Qualität von Batch zu Batch variieren kann. Aliquote repräsentieren also einen Anteil eines Batches der auf dutzende Reagenzbehälter aufgeteilt und gesondert gelagert, bzw. verwendet wird. Aliquote der verschiedenen Batches eines gentechnischen Konstruktes werden in den diversen Kühl- und Frierschränken auf dem Institut gelagert.

## *1.1. Rolle der Gentechnik in der Medizinischen Forschung*

### 1.1.1. Einleitung

Das menschliche Erbgut, die DNA (Desoxyribonukleinsäure; aus dem Englischen "**d**esoxy ribo**n**ucleic **a**cid"), stellt einen Informationsspeicher dar, wie z.B. ein Tonband, auf welchem sich die Information zu dem „Programm des Lebens" befindet. Diese nach rechts gewundene, ähnlich einer Strickleiter aufgebaute Formation, hat auf beiden Seiten der "Holme" jeweils einen Desoxyribose- Baustein und als "Holme" je ein Paar der Basen Adenin, Guanin, Cytosin und Thymin. Hierbei können jeweils nur folgende Basenpaare aufgrund von Wasserstoffbrückenbindungen verbunden sein: Adenin mit Thymin (2 H-Brücken) und Guanin mit Cytosin (3 H-Brücken).Die Information ist in der sequentiellen Abfolge dieser Basen gespeichert. (2).

Die menschliche DNA umfasst ca. 2900 Mega-Basenpaare ($2.9*10^9$), was im "ausgerollten" Zustand einer Länge von ca. 99 Zentimetern entspricht. Die Breite dieser Doppelhelix beträgt ca. zwei Nanometer. Bakterien besitzen eine DNA, welche um einiges kürzer ist (ca. 4000 Basenpaare). Im Unterschied zu der DNA des Menschen, welche aus einem linearen Strang besteht, ist bakterielle DNA zu einem Kreis geschlossen. Sie besitzt also keinen definierten Start- und Endpunkt, im Gegensatz zu der des Menschen. (2)

Da jede kernhaltige Zelle (Nucleus) die DNA für die Herstellung von Proteinen oder, unter anderem, auch zur Zellteilung benötigt, muss diese platzsparend darin verpackt sein. Eine weitere Anforderung an die Verpackungsstrategie muss auch sein, dass jederzeit auf einzelne DNA-Abschnitte zugegriffen werden kann. (2)
Um auf die Basenpaare zugreifen zu können müssen die eingangs erwähnten Wasserstoffbrückenbindungen gespalten werden. Dies geschieht (wenn nicht enzymatisch katalysiert) durch Erhitzen auf ca. 100°C bzw. niedrige Salzkonzentrationen. Damit sich die Einzelstränge wieder zusammenlegen können ("Annealing"), müssen Umgebungsparameter wie Temperatur oder Salzkonzentration auf den ursprünglichen Wert gesetzt werden. (2)
Neben der Rechtswindung der DNA, welche pro Basenpaar um 36° gedreht ist, windet sich die DNA auch noch um sogenannte Histone. Auf jedem dieser Histone befinden sich zwei Windungen der DNA. Diese Histone (auch „Histonproteine" genannt) bestehen aus einem ‚verknäuelten' Protein, welche viele positiv geladene Aminosäuren (wie z.B. Lysin und Arginin) enthält. Da die DNA stark negativ geladen ist, haftet diese an den Histonen und erhält dadurch die dicht gepackte Form. (2, 3)
Der genetische Code, welcher von der DNA abgebildet ist, wird zur Herstellung von Proteinen zuerst abgelesen, um eine sogenannte mRNA (**m**essenger **r**ibo**n**ucleic **a**cid) herzustellen. Bei diesem Transkriptionsvorgang wird ein Strang komplementär zur DNA hergestellt. Das Ergebnis, die mRNA, beinhaltet anstelle der Base Thymin die Base Uracil. Vorteile der mRNA sind, dass für die Herstellung von Proteinen an den Ribosomen nicht immer wieder die DNA entdrillt werden muss, sondern stattdessen viele Kopien des betreffenden Abschnittes in Form der mRNA zur Verfügung stehen. (2)
An den Ribosomen wird, gemäß der Information welche in der mRNA gespeichert ist, das entsprechende Protein hergestellt. Die Information zur spezifischen Aminosäure, welche in die entsprechende Position der wachsenden Peptidkette eingefügt werden soll, liegt in Form sogenannter "Triplets" (Abfolge von drei Basen; auch Codon genannt) vor, was bei vier möglichen Basen pro Position im Triplet theoretisch 64 verschiedene Aminosäuren ermöglicht(4^3 = 64). Da aber nur 20 verschiedene Aminosäuren als Bausteine von Proteinen vorkommen, herrscht hier Redundanz. D.h. es existieren oft mehrere Codons mit unterschiedlichen Basen in letzter Position welche für identische Aminosäuren kodieren. Als Beispiel sei hierfür die Aminosäure Prolin genannt, welche durch 4 verschiedene Triplets, CCT, CCC,

CCA bzw. CCG kodiert werden. Die folgende Abbildung zeigt, welche Aminosäuren aus den Basen-Triplets entstehen: (2)

| Nukleotidbase | | | | | |
|---|---|---|---|---|---|
| Erste | Zweite | | | | Dritte |
| | Uracil (U) | Cytosin (C) | Adenin (A) | Guanin (G) | |
| Uracil (U) | F Phenylalanin (Phe) | S Serin (Ser) | Y Tyrosin (Tyr) | C Cystein (Cys) | U |
| | F Phenylalanin (Phe) | S Serin (Ser) | Y Tyrosin (Tyr) | C Cystein (Cys) | C |
| | L Leucin (Leu) | S Serin (Ser) | Stopp-Codon | Stopp-Codon | A |
| | L Leucin (Leu) | S Serin (Ser) | Stopp-Codon | W Tryptophan (Trp) | G |
| Cytosin (C) | L Leucin (Leu) | P Prolin (Pro) | H Histidin (His) | R Arginin (Arg) | U |
| | L Leucin (Leu) | P Prolin (Pro) | H Histidin (His) | R Arginin (Arg) | C |
| | L Leucin (Leu) | P Prolin (Pro) | Q Glutamin (Gln) | R Arginin (Arg) | A |
| | L Leucin (Leu) | P Prolin (Pro) | Q Glutamin (Gln) | R Arginin (Arg) | G |
| Adenin (A) | I Isoleucin (Ile) | T Threonin (Thr) | N Asparagin (Asn) | S Serin (Ser) | U |
| | I Isoleucin (Ile) | T Threonin (Thr) | N Asparagin (Asn) | S Serin (Ser) | C |
| | I Isoleucin (Ile) | T Threonin (Thr) | K Lysin (Lys) | R Arginin (Arg) | A |
| | Start (Methionin) | T Threonin (Thr) | K Lysin (Lys) | R Arginin (Arg) | G |
| Guanin (G) | V Valin (Val) | A Alanin (Ala) | D Asparaginsäure (Asp) | G Glycin (Gly) | U |
| | V Valin (Val) | A Alanin (Ala) | D Asparaginsäure (Asp) | G Glycin (Gly) | C |
| | V Valin (Val) | A Alanin (Ala) | E Glutaminsäure (Glu) | G Glycin (Gly) | A |
| | V Valin (Val) | A Alanin (Ala) | E Glutaminsäure (Glu) | G Glycin (Gly) | G |

*Abbildung 1: Übersetzung der Basen-Triplets in Aminosäuren (4)*

In der Abbildung 1 sieht man, dass neben den Codons, welche für Aminosäuren kodieren, auch Codons existieren, welche generelle Arbeitsanweisungen darstellen, wie den Beginn des Ablesevorgangs (AUG) und auch dessen Ende (Stoppcodons; UAA, UGA oder UAG). (2)

Zur Verfügung gestellt werden die einzelnen Aminosäuren, welche von den Ribosomen zu Proteinen zusammengestellt werden, durch die Transfer-RNA (**t**ransfer **r**ibo**n**ucleic **a**cid). Diese Moleküle, welche in ihrer 3D-Struktur etwa einem Kleeblatt ähneln, haben an einer speziellen “Andockstelle“ die betreffende Aminosäure gebunden, auf der gegenüberliegenden Seite befindet sich das, dem mRNA Code komplementäre, Anti-Codon für die entsprechende Aminosäure. Da das Anticodon komplementäre Basenfolge zum Codon besitzt, kommt es zur Basenpaarung und die entsprechende Aminosäure ist damit in der Position wo sie dem wachsenden Protein angefügt werden kann. (2)

### 1.1.2. Regulation der Protein-Biosynthese

Da aber nun z.B. eine Hautzelle zum Teil andere Proteine als eine Zelle in der Leber produziert, aber die DNA der beiden Zelltypen identisch ist, muss die Expression der Proteine zellspezifisch reguliert werden. Unterschiedliche Abschnitte der DNA werden differentiell abgelesen bzw. unterdrückt. Mithilfe dieser Drosselung bzw. Zunahme der Proteinsynthese kann sich die Zelle auch an äußere Umstände anpassen und so das richtige Protein zum richtigen Zeitpunkt synthetisieren. (2)
Die Regulation der Proteinexpression wurde anhand des Proteins, welches für den Lactose-Abbau bei E.coli-Bakterien zuständig ist, erstmals erforscht ("Lac Operon"): (2, 3)

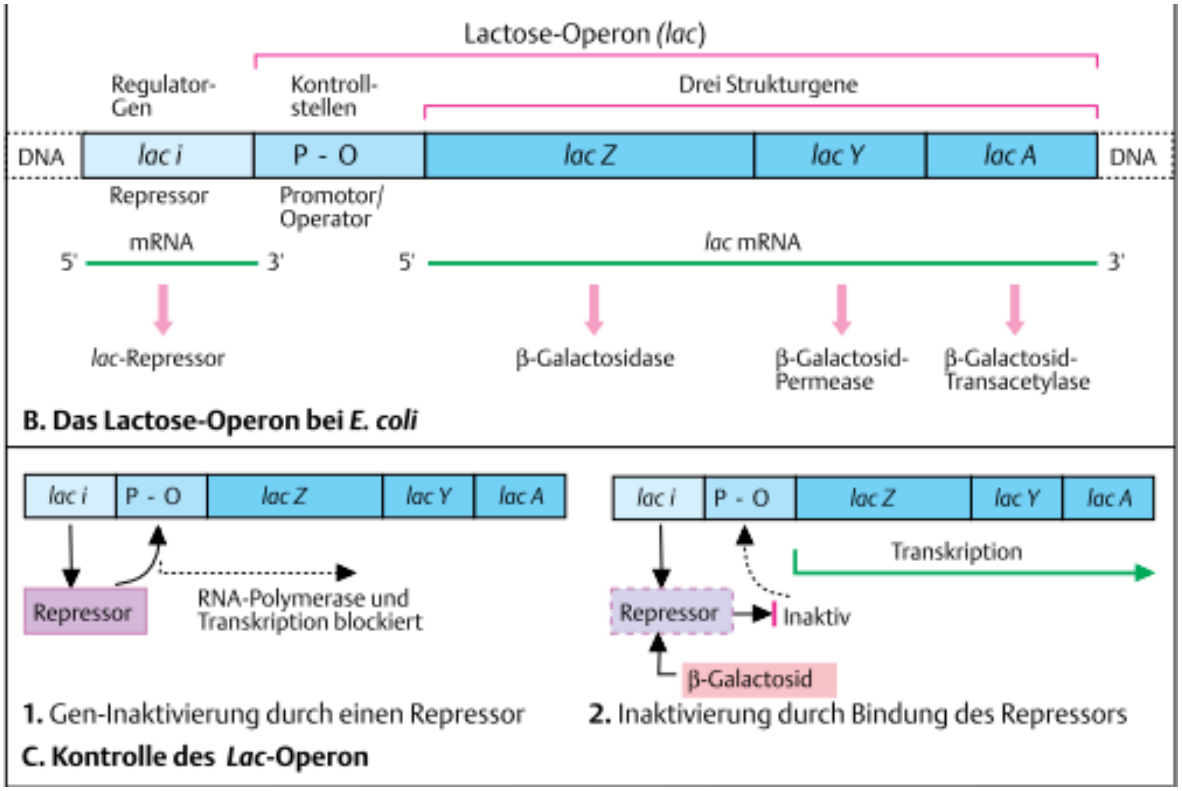



*Abbildung 2: Expression von Laktose-abbauenden Proteinen bei E.coli-Bakterien (3)*

Vor dem Ablesen des DNA-Abschnitts ("upstream"), welcher für die Enzyme des Lactose-Abbaus codiert, existiert ein sequenzspezifischer Abschnitt, der bei niedrigen Lactose Konzentrationen von einem Repressorprotein blockiert wird. Bei steigender Konzentration bindet Lactose an das Repressorprotein, was zu Konformationsänderung und Entblockade führt. Es werden mRNAs transkripiert, welche für die die Enzyme β-Galactosidase, β-Galactosid-Permease und β-Galactosid-Transacetylase kodieren. Mithilfe dieser entstehenden Enzyme kann

Lactose durch das Enzym β-Galactosidase zu Galactose und Glucose umgewandelt werden. (3)

### 1.1.3. Werkzeuge der Gentechnik in der Medizin

#### 1.1.3.1. Restriktionsenzyme

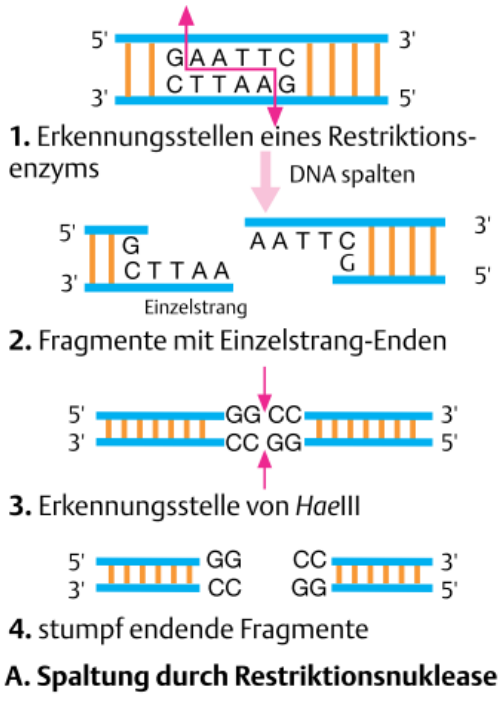
Restriktionsenzyme (“Restriktionsendonucleasen“), dienen dazu eine DNA sequenzspezifisch aufzuschneiden. Hierbei haben diese Enzyme eine gewisse Vorgabe an welcher Sequenz die DNA geschnitten werden soll. Zum Beispiel erkennt das Enzym ‚EcoR1‘ die Sequenz 3‘-GAATTC-5‘ und spaltet die DNA überall dort, wo diese Sequenz vorkommt. Das Enzym spaltet hierbei immer zwischen G und A was zur Folge hat, dass die Enden der beiden aufgetrennten DNA-Abschnitte nicht glatt sind, sondern einen gewissen Überstand aufweisen. (2)

Neben dem benannten Restriktionsenyzm gibt es noch viele weitere Enzyme, welche andere Sequenzen erkennen und auch glatte Doppelstrang-Enden erzeugen können. (2)

Zum Beispiel dient diese Zerstückelung der DNA in mehrere Teile u.a. dazu, die DNA in ‚handliche‘ Stücke zu zerlegen. Dieser Vorgang wird benötigt, da sonst Scherkräfte bei der Isolierung der DNA diese in unkontrollierbar viele Einzelstücke zerlegen würde. Daraufhin wäre die isolierte DNA nicht mehr nutzbar. (2). Restriktionsenzyme ermöglichen also gezielte (“targeted“) Eingriffe in die DNA. Deren Entdeckung und Verfügbarkeit stellt eine Grundvorraussetzung für die Verwendung nahezu sämtlicher gentechnischen Methoden dar. Projektspezifisch werden Sortimente aus dutzenden solcher Restriktionsenzymen in den einzelnen Labors auf Lager gehalten.

Der Name eines Restriktionsenzyms setzt sich aus folgenden Bestandteilen zusammen: (2)

- ***Eco***: Das Bakterium aus welchem das Enzym gewonnen wird - hier steht Eco für Escherichia coli
- ***R***: Dieser Buchstabe steht für den Bakterienstamm aus dem das Enzym gewonnen wurde.
- ***I***: Diese Zahl gibt an welches Enzym verwendet wurde. In diesem Fall wurde das erste gewonnene Restriktionsenyzm verwendet. (2)




| Erkennungssequenz (Schnitt: Symmetrieachse) | Enzym | Mikrobe |
|---|---|---|
| 5' GAA / CTT 3' — 3' CTT / GAA 5' | *Eco*RI | *Escherichia coli* KY 13 |
| 5' GTPy / CAPu 3' — 3' PuAC / PyTG 5' | *Hind*II | *Haemophilus influenzae Rd* |
| 5' AAG / TTC 3' — 3' CTT / GAA 5' | *Hind*III | *Haemophilus influenzae Rd* |
| 5' GTT / CAA 3' — 3' AAC / TTG 5' | *Hpa*I | *Haemophilus parainfluenzae* |
| 5' CC / GG 3' — 3' GG / CC 5' | *Hpa*II | *Haemophilus aphrophilus* |
| 5' GG / CC 3' — 3' CC / GG 5' | *Hae*III | *Haemophilus aegyptius* |

**B. Beispiele für Restriktionsenzyme**

*Abbildung 3: DNA-Spaltung durch Restriktionsenzyme (3)*

### 1.1.3.2. Ligasen

Ligasen haben den Zweck, dass DNA Fragmente wieder zu einem DNA-Stück zusammengesetzt werden. Hierbei katalysieren sie die Entwicklung von Phosphordiesterbrücken zwischen den Nucleinsäure-Ketten. (2)

### 1.1.3.3. Vektoren

Vektoren werden benötigt um genetische Informationen, wie z.B. eine DNA, in eine Zelle einzuschleusen. Dort soll diese eingeschleuste Information – wie z.B. bei Bakterien eine Ampicillin-Resistenz - exprimiert werden. Neben dem Einschleusen des Erbguts soll der Vektor auch dafür sorgen, dass die eingeschleuste DNA vervielfältigt wird. Aus einem Molekül im Reagenzglas gentechnisch angefertigter DNA können so multiple Kopien im mg Bereich angefertigt werden. (2).

Zum Einsatz kommen für diesen Zwecke oft Plasmide oder Viren. Diese müssen vor ihrem Start noch an ihre Aufgabe angepasst werden. In diese Vektoren muss die zu übertragende DNA mithilfe von Werkzeugen wie Restriktionsenzymen und Ligasen enzymatisch eingebaut werden. Dies geschieht entweder durch Insertion (Ergebnis ist hier ein Insertionsvektor) oder durch Substitution (Ergebnis ist hier ein

Substitutionsvektor). Bei der Insertion wird die zu übertragende DNA der DNA des Vektors hinzugefügt – die ursprüngliche DNA bleibt erhalten. Bei der Substitution wird ein Teil der DNA des Vektors mithilfe von Restriktionsenzymen entfernt und durch die einzuschleusende DNA (auch „Passagier-DNA“ genannt) ersetzt. Um nun diese im Wirt mit dessen DNA verbinden zu können, werden wiederum Restriktionsenzyme verwendet. Diese schneiden neben der Passagier-DNA auch die DNA des Wirts. Beim Wiederzusammenfügen der DNA-Fragmente vermischen sich nun, aufgrund der gleichen Enden aller DNA-Fragmente, diese wieder und die Passagier-DNA findet sich zwischen der ‚normalen‘ Wirts-DNA wieder und kann von dort aus exprimiert werden. (2)

Es gibt zwei Arten von Vektoren: Klonierungsvektoren und Expressionsvektoren. Die Hauptaufgabe der Klonierungsvektoren besteht darin, die Passagier-DNA in der Zielzelle zu vermehren. Anders hier die Expressionsvektoren: Diese sind dafür ausgelegt, dass die DNA nicht nur vermehrt, sondern im, durch die Transformation gentechnisch veränderten Organismus (“GVO“), Proteine exprimiert werden, die dann aufgereinigt und verwendet werden können. (2)

#### 1.1.3.4. **Primer**

Primer, welche aus kurzen RNA-Stücken bestehen, werden verwendet um die Startposition bei der Replikation von DNA zu markieren. Die Herstellung der Primer, welche eine Länge von durchschnittlich zehn bis zwölf Nukleotiden haben, übernimmt das Enzym *Primase*. Neben der Herstellung übernimmt dieses Enzym die Anlage der Primer an die zu replizierende, bereits entdrillte, DNA. Nach der Anlage kann die Verdopplung der DNA von 5‘ nach 3‘ erfolgen. (5)

Nach der Verdopplung werden mittels der DNS-Polymerase die Primer wieder aus der DNA entfernt und die entstandenen Lücken mittels DNS-Ligase verschlossen. (5)

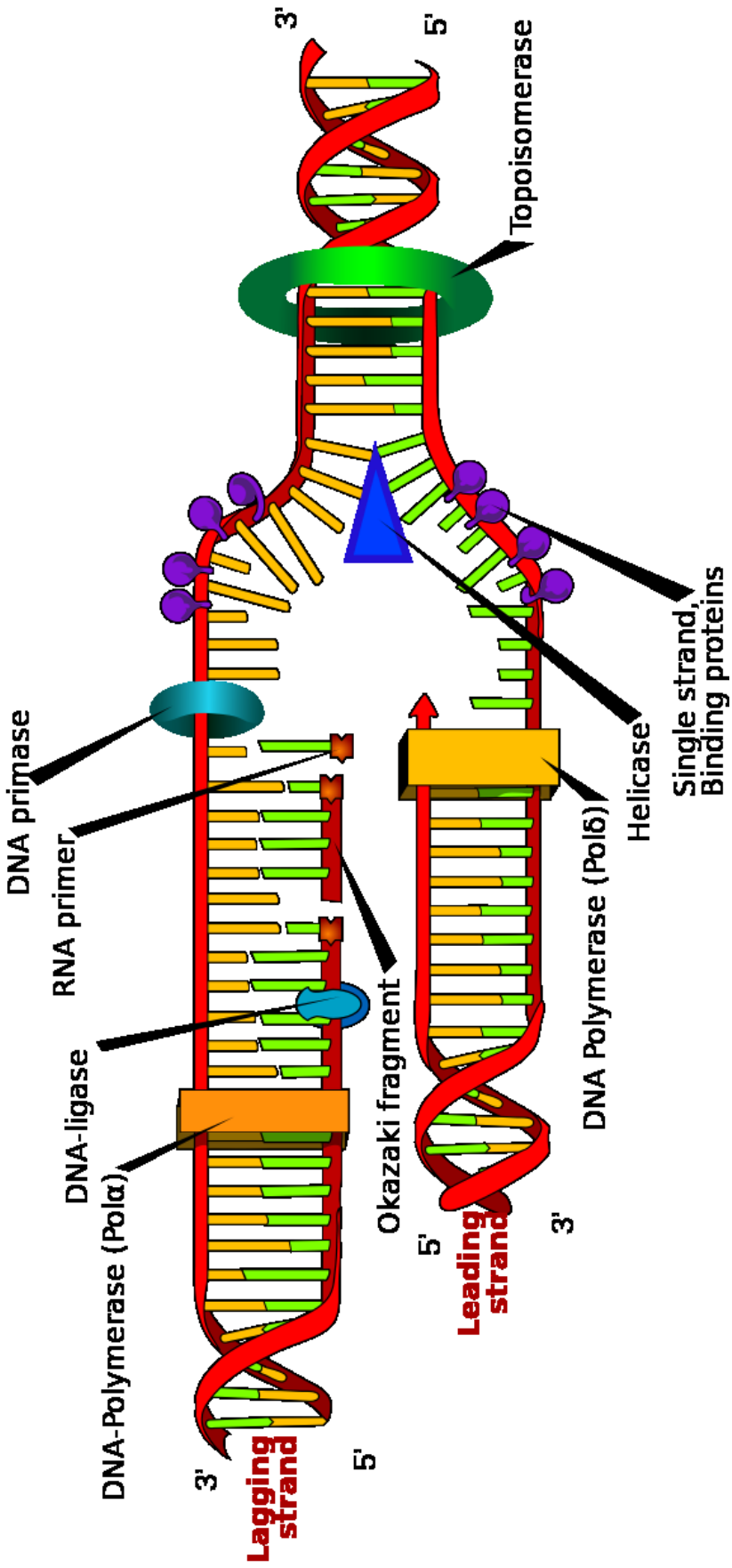



*Abbildung 4: Verwendung eines RNA-Primers zur DNA-Duplizierung (1)*

#### 1.1.3.5. Multiple Cloning Site

*„Ein Polylinker oder auch Multiple Cloning Site (MCS) ist ein künstlich geschaffenes DNA-Oligonukleotid aus rund 50bp, welches in ein Plasmid eingesetzt wird und dessen Sequenz direkt hintereinander verschiedene Restriktionsschnittstellen für Restriktionsendonukleasen enthält.“ (6)*

Mithilfe einer Multiple Cloning Site kann man sicherstellen, dass die einzuschleusende DNA immer an der gewünschten Stelle eingebaut wird. Dies geschieht durch die beigefügten Restriktionsenzyme, welche sequenzspezifisch passende Andockstellen für das einzufügende Insert generieren. Hierbei ist darauf zu achten, dass die DNA-Abschnitte, welche von den Restriktionsenzymen erkannt werden, sich nur innerhalb des Polylinkers befinden. Falls dies aber doch der Fall sein sollte, müsste man die DNA-Abschnitte mithilfe von Mutationen entfernen. (6)

#### 1.1.3.6. Gentechnisches Konstrukt

Nachdem z.B. in ein Phagen, welcher als Vektor verwendet wird, eine Gensequenz eingebracht wurde, wird dieses als Konstrukt bezeichnet. Neben solchen Phagen, welche Bakterien befallen, gibt es unter anderem Plasmide, Cosmide und Phagemide, welche als Konstrukte eingesetzt werden können. (7)

## *1.2. Problemstellung*

Bis vor der Fertigstellung dieses Software-Projekts wurden die Daten und Eigenschaften der gentechnischen Konstrukte handschriftlich auf Zetteln, in Microsoft Excel-Arbeitsmappen inklusive Verweisen auf andere Tabellenblätter, mehreren Microsoft-Word-Dokumenten und diversen Ordnern am Computer und in Netzlaufwerken gespeichert. Es existierte keine Indexierung der Konstrukte um effektiv nach einem speziellen Konstrukt suchen zu können. Um ein spezielles Konstrukt finden zu können, musste man entweder die Nomenklatur der Dateinamen kennen oder das direkte Gespräch mit einem Mitarbeiter des Instituts suchen, welcher Durchblick und Erfahrung mit dieser komplexen Form der Dokumentation und Archivierung besitzt.

## 1.3. Ansatz der Problemlösung

Um die Verwaltung der gentechnischen Konstrukte in geordnete Bahnen leiten zu können, wurde in mehreren Zusammenkünften mit Mitarbeitern des Instituts für Biophysik analysiert, aus welchen Bestandteilen solche Konstrukte bestehen und wie diese aufgebaut sind, um deren Datenstrukturen in einer relationalen Datenbank abbilden zu können.

Es war auch der Wunsch, dass die entstehende Software nach dem Client-Server-Model aufgebaut ist und eine zentrale Datenspeicherung erfolgt. Der Zugriff auf den Server soll nur über das lokale Institutsnetzwerk erfolgen können.

Des Weiteren sollte das Augenmerk auf kostenfreie Software für die Entwicklung der Software sowie der Verwaltung der Daten im Hintergrund gelegt werden. Da über die Medizinische Universität Graz für den Institutsgebrauch mit mehreren Software-Firmen Verträge abgeschlossen wurden, hat sich hier kein Problem gestellt.

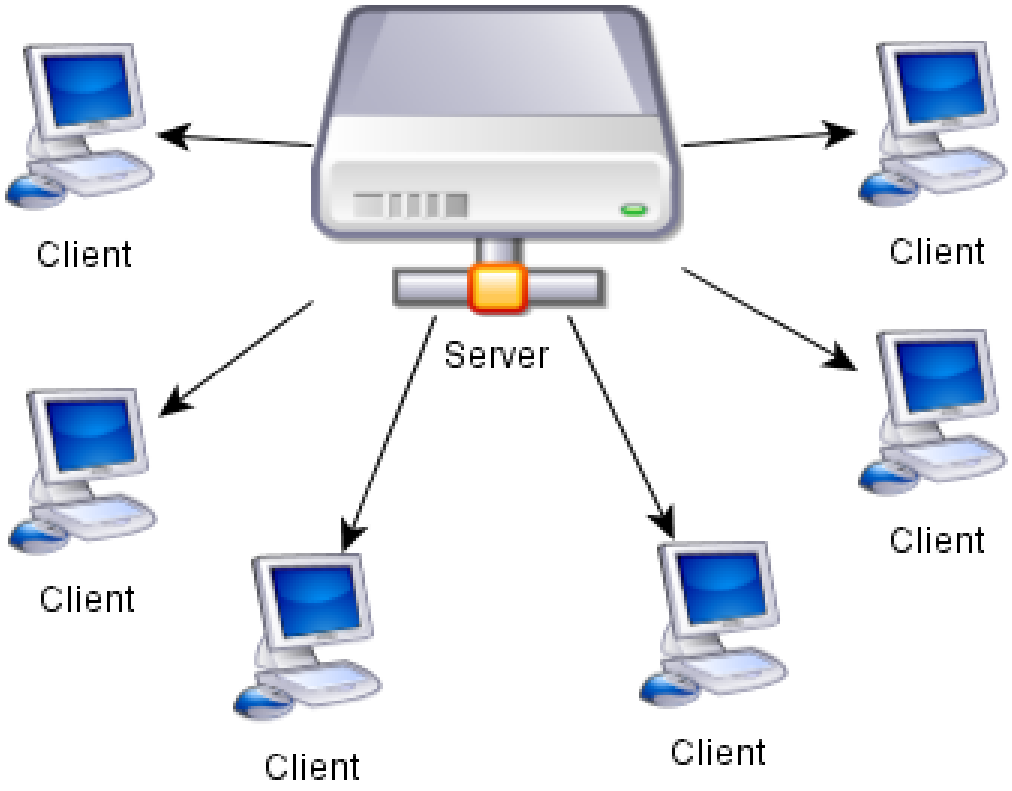



*Abbildung 5: Typische Client-Server-Umgebung in einem Netzwerk (8)*

# 2. Material und Methoden

Um solch ein Softwareprojekt realisieren zu können muss die neue, programmierte Software mit der bereits bestehende Software kompatibel sein.

Deswegen wurde als Entwicklungsumgebung das Programm „Visual Studio 2013“ von Microsoft eingesetzt. Die Software wurde in der Programmiersprache C# mit der Laufzeitumgebung .NET in der Version 4.5.1 erstellt. Die entstehenden Daten, welche von den Instituts-Mitarbeitern eingegeben werden, werden wie folgt abgespeichert:

- Dateien und Ordner, welche im Rahmen der Dokumentation anfallen, werden im Hintergrund auf einen FTP Server hinterlegt
- Daten und Informationen, aus welchen die Konstrukte bestehen, werden auf einem SQL Server gespeichert

## *2.1. FTP Server*

*“The objectives of FTP are 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, […]” (9)*

Ein FTP Server stellt über ein Netzwerk Dateien und Ordner zur Verfügung und ermöglicht auf diesem selbst Dateien und Ordner abzulegen. Für dieses Projekt wurde ein FTP Server benötigt, da die Dokumentation zu den einzelnen Objekten wie Vektoren, Spezies, Konstrukte usw. nicht auf dem SQL Server gespeichert werden sollen. Dies würde zu unnötigem Verbrauch von freiem Speicherplatz in der Datenbank führen. Der SQL Server 2014 von Microsoft in der Express-Version, welche für alle Anwendungsmöglichkeiten gratis verfügbar ist, besitzt eine maximale Datenbankgröße von 10 Gigabyte. (10)

Da in der Regel die Dokumente (Bilder, Microsoft Word-Dokumente, …) den meisten Speicherplatz benötigen, wurde diese Datenmenge auf einen FTP Server (welcher nur durch das Datenvolumen der Festplatte begrenzt wird) ausgelagert.

Für dieses Projekt wurde der FileZilla FTP Server von Tim Kosse in der Version 0.9.56.1 verwendet. Dieser kann kostenlos unter der Open-Source-Lizenz GNU General Public License (GPL) version 2 über die Projekthomepage[1] heruntergeladen werden. Des Weiteren kann man bei der Installation angeben,

[1] https://filezilla-project.org/download.php?type=server (Stand 29.04.2016)

dass der FTP Server als Service im Hintergrund läuft und somit sofort startet, wenn der Computer hochgefahren wird. Zusätzlich kann der FTP Server über ein mitgeliefertes Interface konfiguriert werden. So kann man mit diesem Interface unter anderem folgende administrative Tätigkeiten ausführen: (11)

- Benutzer erstellen, Passwörter zuweisen
- Maximale Geschwindigkeit der Datenübertragung festlegen
- IP Filter setzen – somit können z.B. gezielt einzelne Computer im Netzwerk von der Benutzung des FTP Servers ausgeschlossen werden
- Das Arbeitsverzeichnis setzen, inklusive der Zugriffsbeschränkungen
- Zugriffe auf den FTP Server überwachen

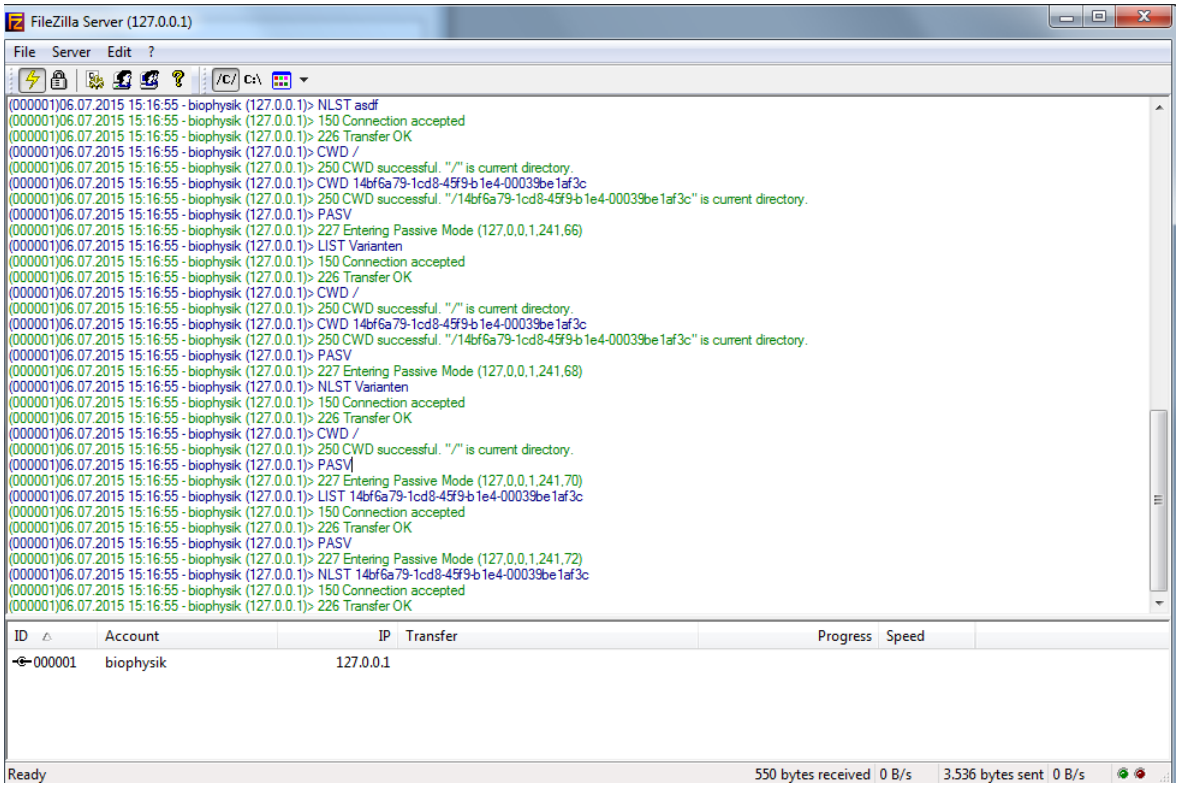



*Abbildung 6: Das FileZilla Server Interface während einer Abfrage*

Um auf die Daten, welche auf dem FTP Server abgelegt sind, ohne die in dieser Arbeit beschriebene Verwaltungssoftware zu verwenden zugreifen zu können, kann man das Arbeitsverzeichnis entweder am Computer, auf welchem der FTP Server läuft öffnen oder mit einem FTP Client (z.B. dem FileZilla FTP Client – kostenlos verfügbar über dessen Projekthomepage[2]), auf die Daten zugreifen.

[2] https://filezilla-project.org/ (Stand 21.09.2015)

Wie läuft nun solch ein Kommunikationsprozess zwischen einem FTP Server und einem FTP Client ab? Zuerst werden nach dem initialen Verbindungsaufbau vom Client zum Server mittels Verwendung der Adresse und des Ports zwei TCP-Verbindungen zwischen Client und Server aufgebaut. Einerseits eine Kontrollverbindung, um Befehle senden zu können, andererseits eine Datenverbindung, um Dateien und Ordner übertragen zu können. Werden nun mehrere Dateien zwischen Server und Client übertragen, wird keine zusätzliche Kontrollverbindung (diese existiert nur einmal zwischen Server und Client), aber für jeden parallelen Dateiversand weitere Datenverbindungen aufgebaut. Bei vielen gleichzeitigen Dateiübertragungen und womöglichem Zugriff mehrerer Clienten auf einen einzelnen Server muss sich dieser auch merken können, welcher Datenstream zu welchem Client gehört. Dies ist vor allem wichtig, wenn sich z.B. ein Client während dem Datenversand abmeldet – hier müssen dann auch die zum Client gehörigen Datenverbindungen unterbrochen werden. (12)

In der Kontrollverbindung werden für den Menschen lesbare FTP-Befehle verwendet. Ein kompletter Befehl beginnt mit vier Zeichen für den Befehl an sich, gefolgt von einem Parameter bzw. Argument, z.B. bei einer Auflistung der Dateien in einem Ordner, mit dem Verzeichnispfad. (12)

Bei der Anmeldung werden folgende Befehle verwendet:

- „USER *Benutzername“* – um den Benutzernamen zu übertragen
- „PASS *Passwort*“ – um nach dem Benutzernamen das Passwort an den Server zu senden (12)

Nach der Anmeldung können, unter anderem, folgende FTP-spezifische Befehle verwendet werden:

- „LIST“ – listet alle Dateien und Ordner des aktuell verwendeten Verzeichnisses auf. Diese Übertragung geschieht über eine Datenverbindung.
- „RETR *Filename*“ – Ruft eine Datei über eine Datenverbindung vom FTP Server ab.
- „STOR *Filename*“ – Überträgt eine lokale Datei über eine Datenverbindung in das aktuelle Verzeichnis des FTP Servers. (12)

Damit der FTP Client erfährt, ob die Übertragung erfolgreich bzw. ob der aktuell gesendete Befehl in Ordnung war, antwortet der FTP Server mit einer dreistelligen Nummer und einer optionalen Antwort. (12)

## *2.2. Microsoft SQL Server*

*„Der Microsoft SQL Server (auch kurz MSSQLServer) ist ein relationales Datenbankmanagementsystem von Microsoft.“ (13)*

### 2.2.1. Einführung

Mit SQL („Structured Query Language“) kann man Datenbanken erstellen und diese auch verwalten. Als Datenbank kann man eine Ansammlung von Tabellen verstehen, welche wie folgt aufgebaut sind:

- In den einzelnen Spalten einer Tabelle werden Name und Art der Daten, welche in dieser Spalte hinterlegt werden können, definiert. So kann man angeben, dass z.B. nur Zahlen oder nur Zeitangaben in einer entsprechenden Spalte gespeichert werden können. Des Weiteren kann man angeben, ob ein NULL (gleichzusetzen mit „nicht angegeben“)-Wert für diese Spalte möglich ist.
- Jede Zeile (auch „Tupel“ oder „Record“ genannt) beinhaltet einen Datensatz. Dieser kann nur und muss aus den Eigenschaften, die über die Spalten definiert wurden, bestehen.

Um nun die Daten in einer konsistenten Form ablegen zu können, müssen deren Beziehungen untereinander normalisiert werden. Dies bedeutet, dass Redundanzen in den Daten eliminiert werden. Dieser Normalisierung hat sich E.F. Codd im Jahre 1972 angenommen und hat Regeln aufgestellt in welcher Reihenfolge diese Normalisierung von Daten in Tabellen stattfinden soll. Neben den wichtigsten drei Normalisierungsschritten existieren noch weitere Schritte, welche aber nur für spezielle Anwendungsgebiete wichtig sind: (14)

Bevor man sich an die Schritte der Normalisierung seiner Daten wagt, benötigt man für jede Tabelle einen „unique identifier“, welcher den Datensatz in der Tabelle einmalig macht. Natürlich darf diese Spalte keinen NULL-Wert zulassen. Zur Verwendung kommen hier z.B. eine GUID, eine fortlaufende Nummer, eine Spalte oder auch die Kombination mehrerer Spalten einer Tabelle. (14)

Um hier den Rahmen dieser Arbeit nicht zu sprengen hier nur die Regeln, welche nach der ersten Normalisierungsform gelten müssen:

- Jede Zelle in einer Tabelle darf nur einen einzigen Wert beinhalten
- Jede Zeile muss die gleichen Attribute wie jede andere Zeile beinhalten

- Eine komplette Zeile darf nicht in jedem Attribut in einer anderen Zeile gleichen (14)

### 2.2.2. Entscheidung für den SQL Server von Microsoft

Neben dem SQL Server von Microsoft existieren weitere relationale Datenbanken wie zum Beispiel MySQL, PostgreSQL, Oracle 10g Express, usw. um hier nur einige zu nennen.
Es wurde der SQL Server 2014 von Microsoft gewählt, da dieser von der Entwicklungsumgebung Microsoft Visual Studio 2013 nahtlos ohne weitere Add-Ons verwendet werden kann. Es wurde auch die Express-Version dieses SQL-Servers auf dem Installationsmedium der Entwicklungsumgebung mitgeliefert. Ein Umstieg auf eine leistungsstärkere Version des MSSQL Servers wäre natürlich auch ohne größeren Aufwand möglich. Die 10 Gigabyte-Beschränkung der Express-Version des MSSQL Server 2014 stellt aufgrund der Auslagerung von großen Dateien und Ordner auf den FTP Server ebenso keine Begrenzung dar. Von den Mitarbeitern des Instituts für Biophysik wurde geschätzt, dass maximal 500 Megabyte an Daten in den nächsten Jahren anfallen würden. Somit sollte die Express-Version dieses MSSQL-Servers voll und ganz genügen, ein Upgrade auf eine leistungsstärkere, kostenpflichtige, Version des MSSQL Servers ist aber jederzeit möglich. (10)
Auch kann mit dem mitgelieferten SQL Server 2014 Management Studio in einer grafischen Oberfläche auf den MSSQL Server zugegriffen werden. Mit dieser GUI konnten mittels „Drag and Drop“ Tabellen und Relationen erstellt werden. Des Weiteren war dies kein Front-End für eine Ausgabe-Konsole (wie z.B. beim MySQL-Server bekannt), sondern man konnte mit diesem Management-Programm direkt auf die Datenbank zugreifen und Hersteller-spezifische Funktionen und Performance-Verbesserungen ausführen, welche z.B. vom SQL:2006-Standard nicht vorgesehen sind. (14)
Auch wird der Zugriff auf den MSSQL-Server vom .NET-Framework ohne weitere Add-Ons unterstützt, was zur Folge hat, dass weniger Programmier-Arbeit in die Umsetzung dieser Verwaltungssoftware in diesem Teilbereich investiert werden musste.

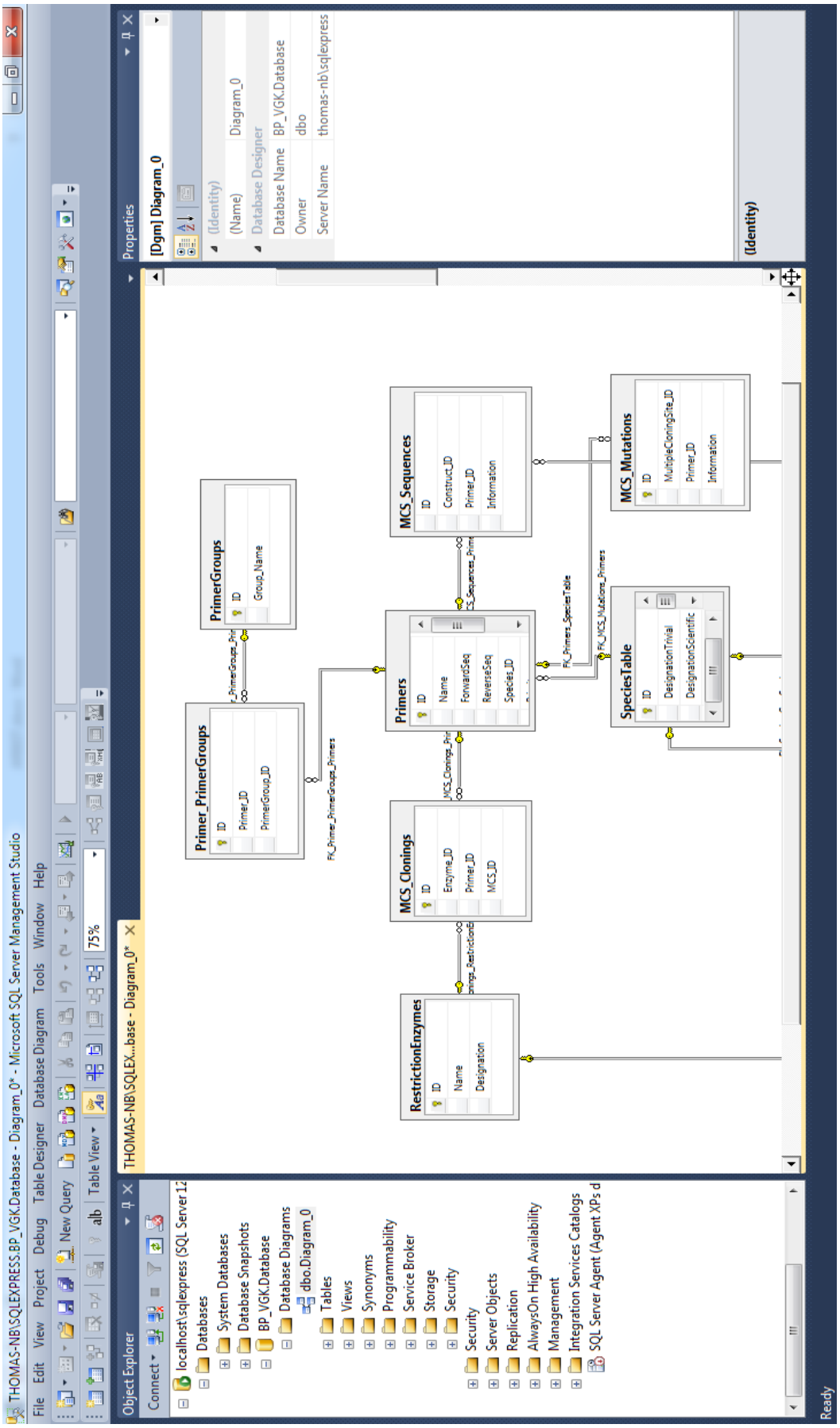



*Abbildung 7: Das Microsoft SQL Server Management Studio während dem Anzeigen eines Ausschnittes der Tabellen-Beziehungen der verwendeten Datenbank*

Ein weiterer Vorteil vom MSSQL Server von Microsoft ist die Tatsache, dass sogenannte „Query Notifications“ vom SQL Server ausgesendet werden können. Eine solche Query Notification kann für ein SQL-Statement angelegt werden. Wenn sich in den Daten, die dieses SQL-Statement liefert, eine Änderung ergibt wird der Client mit einer Mitteilung, ähnlich einer Push-Nachricht, benachrichtigt. Somit muss die Datenbank nicht unnötigerweise in einem bestimmten Intervall immer wieder abgefragt werden, wie der aktuelle Stand der Daten einer Tabelle ist, sondern man bekommt dies von der Datenbank selbst mitgeteilt. Diese Hilfestellung seitens der Datenbank erspart neben der Rechenleistung an den Client auch einiges an Netzwerkauslastung, da nur Daten über das Netzwerk gesendet werden, wenn sich etwas in der Datenbank ändert.

## *2.3. Microsoft Visual Studio*

Mit dem Visual Studio 2013 von Microsoft lassen sich neben ‚normalen‘ Software-Projekten auch Webseiten und Teamprojekte erstellen. Für Software-Projekte kann man unter anderem folgende Programmiersprachen verwenden: C#, Visual Basic, J# oder auch Python. Für die Erstellung von Webseiten kann die Programmiersprache ASP.NET verwendet werden.

Bei Teamprojekten wird ein ‚Team Foundation Server‘ benötigt, welcher den Source Code der Software zentral lagert und diesen für alle Teammitglieder zur Verfügung stellt.

Um ähnlich zu dem Team Foundation Server von Microsoft auf ältere Versionen des Quellcodes zugreifen zu können, wurde für dieses Projekt der SVN Server von der Apache Software Foundation verwendet, welcher auf einer Synology DiskStation läuft. Um mit dieser Quellcodeverwaltung arbeiten zu können, wurde das Add-On VisualSVN verwendet, welches für den nicht kommerziellen Gebrauch kostenlos ist und von VisualSVN Limited entwickelt wurde.

# 3. Resultate

Die Software zur Verwaltung der gentechnischen Konstrukte wurde in einer Client-Server-Architektur realisiert. Auf der Seite des Clients, befindet sich die entwickelte Client-Software, welche in der Programmiersprache C# - aufbauend auf das .NET-Framework – entwickelt wurde. Auf der Seite des Servers wurde der Microsoft SQL Server 2014 und zur Verwaltung der Dokumentation der einzelnen Datensätze der FileZilla FTP Server in der Version 0.9.56.1 installiert.

Das Client-Programm selbst befindet sich inklusive Quellcode und aller verwendeten Programme auf der beigelegten DVD.

## *3.1. Programmstruktur*

Die Client-Anwendung basiert auf drei Klassenbibliotheken und einer ausführbaren Anwendungsdatei. Außerdem wird eine Einstellungsdatei benötigt um diverse Einstellungen abzuspeichern.

Folgende Dateien müssen sich neben der Anwendungsdatei – „BP_VGK.VerwaltungGentechKonstrukte.exe“ in diesem Ordner befinden:

- BP_VGK.Common.dll
  - Stellt allgemeine Prozeduren und Informationen zur Verfügung
- BP_VGK.Data.FTP.dll
  - Wird als ‚Connector‘ zum FTP Server benötigt
- BP_VGK.Data.MSSQL.dll
  - Stellt die Daten vom SQL Server für die Anwendung bereit
- Settings.ini
  - In dieser Datei werden veränderbare Eigenschaften für die Verwendung des Programms hinterlegt. Diese Daten können vom Benutzer bei Bedarf verändert werden.

Für die Erstellung der Verbindung zum FTP-Server wurde auf eine bereits vorgefertigte C#-Klasse zurückgegriffen. Nach diversen Anpassungen der Software konnte diese für dieses Projekt verwendet werden. Der verwendete Quellcode steht unter der CPOL-Lizenz 1.02 und kann laut dieser Lizenz für solch ein Software-Projekt verwendet werden. (15, 16)

### 3.1.1. Inhalt der Einstellungsdatei „Settings.ini“

Die Einstellungsdatei „Settings.ini“ beinhaltet vier Abschnitte mit dazugehörigen Schlüssel-Wert-Paaren.

**Abschnitt SQLConnection:**

- *ConnectionString*: Hier wird die Verbindungszeichenfolge für die Verbindung zum SQL-Server hinterlegt. Folgende Platzhalter werden zur Laufzeit mit den vorgegebenen Zeichenfolgen ersetzt:

| Platzhalter | Wird zur Laufzeit ersetzt durch |
|---|---|
| {{MSSQLSERVERADDRESS}} | IP Adresse bzw. DNS-Name des SQL-Servers |
| {{DATABASENAME}} | Name der Datenbank |
| {{USERID}} | Benutzername zur Anmeldung am SQL-Server |
| {{PASSWORD}} | Passwort zur Anmeldung am SQL-Server |

**Abschnitt Formatting:**

- *BatchesDateFormat*: Da bei der Verwaltung der Batches das Datum der Erstellung mitgespeichert und angezeigt wird, kann mithilfe dieses Schlüssel-Wert-Paares die Formatierung dieses Datums personalisiert werden. Hierbei kann jedes Format, das bei der verwendeten Klasse unterstützt wird, verwendet werden. [3]

**Abschnitt Settings:**

- *ShowKeysInsteadOfToolTips*: Um während der Laufzeit herauszufinden, zu welchem Steuerungselement Tooltips hinterlegt werden können bzw. wie der Schlüssel zu einem Element heißt, kann mithilfe dieser Funktion zwischen der Anzeige des hinterlegten Tooltips und dem verwendeten Schlüssel für dieses Element umgeschaltet werden.

  Wird hier die Zahl „1“ als Wert hinterlegt, wird beim nächsten Programmstart der Schlüssel zu dem verwendeten Tooltip als Tooltip angezeigt. Bei jedem anderen verwendeten Wert wird der normal hinterlegte Tooltip angezeigt.

[3] https://msdn.microsoft.com/en-us/library/system.windows.forms.datetimepicker.customformat%28v=vs.110%29.aspx (Stand 15.02.2016)

**Abschnitt Login:**

- *ServerAddress*: Da zumeist ein Client nur auf einen Server zugreift – und sich diese Adresse zumeist nicht ändert, wurde folgende Funktion eingeführt. Nach einem erfolgreichen Login wird die verwendete Server-Adresse hinter diesem Schlüssel hinterlegt und diese Adresse beim nächsten Neustart des Clients automatisch im Feld "Server Address“ im Anmeldefenster eingefügt.

## *3.2. Datenbankstruktur*

Die SQL-Datenbank umfasst insgesamt achtzehn Tabellen und neun Views. Jedem Datensatz wurde bei der Erstellung eine eindeutige GUID zugewiesen. Diese ist dank ihrer Länge von 128 Bit und zufälliger Generierung durch den SQL Server selbst einzigartig. (17)
Außerdem besitzt jede Tabelle, außer DBVersion, MultipleCloningSites, PrimerGroups, ToolTips und solche die eine m:n-Beziehung auflösen, eine ‚Note‘-Spalte, um zu jedem Datensatz eine kurze Beschreibung hinterlegen zu können. Folgende Tabellen wurden zur Verwaltung der Datensätze erstellt:

- Batches
    - Diese Tabelle enthält die Daten in welchem „Freezer“ sich ein gentechnisches Konstrukt am Institut befindet. Außerdem findet sich hier die Angabe, welchem Benutzer dieser Batch zugeordnet ist.
- Constructs
    - In dieser Tabelle befinden sich folgende Daten zu den erstellten Konstrukten: Erstell- und Änderungsdaten, Zuordnung des Vektors und Daten zur Sequenzierung des gentechnischen Konstrukts.
- DBVersion
    - Da die Datenbank über die Clients aktualisiert werden kann (z.B. wenn ein neues Feature der Software eine Datenbank-Aktualisierung benötigt), wird eine Versionsnummer der Datenbank in dieser Tabelle abgelegt. Somit werden allfällige Aktualisierungen nur einmal in die Datenbank eingespielt.
- GeneProducts
    - Neben dem Namen eines Genprodukts wird der Verweis auf das erzeugende Gen in dieser Tabelle hinterlegt.

- GeneSpecies_Table
    - Da zwischen den Tabellen ‚GeneTable' und ‚SpeciesTable' eine m:n–Beziehung besteht (Ein Gen kann mehreren Spezies sowie eine Spezies mehreren Genen zugeordnet sein) wird diese Beziehung mithilfe dieser Tabelle aufgelöst.
- GeneTable
    - Die Namen der Gene bzw. deren Text-Informationen werden in dieser Tabelle abgebildet.
- MCS_Clonings
    - Diese Tabelle dient wie die ‚GeneSpecies_Table' zur Auflösung einer m:n-Beziehung zwischen mehreren Tabellen. Da eine Klonierung in der Multiple Cloning Site einen Primer und ein Restriktionsenzym beinhaltet, wurde die Auflösung mithilfe dieser Tabelle realisiert. Außerdem wurde zur Festhaltung der Position der Klonierung eine Positionsnummer hinterlegt.
- MCS_Mutations
    - Da in jeder Multiple Cloning Site mehrere Primer inklusive eines Texts hinterlegt werden können, wird diese Information in dieser Tabelle abgebildet.
- MCS_Sequences
    - Ähnlich wie in der ‚MCS_Mutations' – Tabelle werden hier Primer und eine Information zu einer Multiple Cloning Site zugeordnet.
- MultipleCloningSites
    - Zu jeder Multiple Cloning Site werden folgende Informationen hinterlegt: ID des zugehörigen gentechnischen Konstrukts, das verwendete Gen, die dazugehörige Spezies, welche Basenpaare zur Sequenzierung verwendet wurden und die ID des verwendeten Genprodukts.

- Primer_PrimerGroups
    - Da sowohl ein Primer mehreren Primer-Gruppen als auch eine Primer-Gruppe mehrere Primer angehören, wird diese m:n-Beziehung in dieser Tabelle aufgelöst.
- PrimerGroups
    - Es existieren drei Primer-Grupen: „Cloning“, „Mutation“ und „Sequenzing“. Es wird beim Programmstart überprüft, ob diese mit einer festgelegten ID in dieser Tabelle existieren. Falls dem nicht so ist werden diese in dieser Tabelle hinterlegt.
- Primers
    - Neben dem Namen des Primers wird in dieser Tabelle die zugehörige Spezies verlinkt, die Forward- und die Reverse-Sequence zu jedem Primer hinterlegt. Des Weiteren existiert eine ‚Priority‘-Spalte, welche dazu dient, dass der Wildtyp-Primer in den Auflistungen immer an erster Stelle angezeigt wird Diesem wurde die Priorität 1 hinterlegt, den restlichen Primer wird hierbei die Zahl 100 zugeordnet.
- RestrictionEnzymes
    - Die Verwaltung der Restriktionsenzyme beinhaltet neben dem Namen des Enzyms eine kurze Beschreibung zu diesen.
- SpeciesTable
    - Neben einer Trivial-Bezeichnung wird hier auch die wissenschaftliche Bezeichnung der Spezies hinterlegt.
- ToolTips
    - Hinter verschiedenen Benutzersteuerelementen im Client-Programm können Pop-ups hinterlegt werden, welche die Funktion dessen näher erläutern. Diese Zuordnung wurde mittels Schlüssel-Wert-Paar realisiert. Der Schlüssel enthält neben dem Namen des Tabs eine Kurzform der Art des Benutzersteuerelements sowie eine Kurz-Beschreibung Elements selbst. Der Wert dieses Paars enthält den Text, der angezeigt werden soll, wenn der Benutzer mit der Maus über dem Element steht.

- Users
    - In dieser Tabelle werden die Benutzer des Programms verwaltet. Neben dem Benutzernamen wird hier das Passwort in MD5-verschlüsselter Form hinterlegt. Außerdem wird abgespeichert, ob der Benutzer-Account aktiviert und ob dieser Administratorrechte besitzt.
- Vectors
    - Hier werden die Daten zu den Vektoren abgespeichert. Neben dem Namen des Vektors kann auch noch eine Text-Dokumentation hinterlegt werden.

Des Weiteren werden Views verwendet, um zusammengefügte Daten aus den Tabellen einfacher auszulesen. Diese virtuellen Tabellen, welche durch SQL-Statements erstellt werden, werden für die Anzeige der Daten im Client-Programm benötigt. (18)

- Batches_Constructs_Owner_View
    - Dieser View setzt die Daten aus den Tabellen ‚Batches', ‚Constructs' und ‚Users' zusammen, damit alle Batches und deren zugehörigen Konstrukte, sowie deren Ersteller einfacher angezeigt werden können.
- Constructs_Vectors_MCS_View
    - Als Grundlage wird der ‚Constructs_Vectors_View – View verwendet um die zugehörigen Vektoren zu den Konstrukten anzuzeigen. Mit diesem View wird die Anzahl an Multiple Cloning Sites, die ein Konstrukt beinhaltet, zur Ausgabe hinzugefügt.
- Constructs_Vectors_View
    - Hier werden die Konstrukte zusammen mit den Namen der verwendeten Vektoren in einer virtuellen Tabelle angezeigt.
- GeneProducts_Gene_View
    - Dieser View liefert die Namen der Gen-Produkte, die den Genen zugeordnet sind.
- GeneSpecies_Species_View
    - Ausgehend von der ‚GeneSpecies_Table' - Tabelle werden die dazugehörigen Namen der Spezies abgerufen.

- Primer_MCS_Clonings_View
    - Es werden die Daten der Primer, auf die in der ‚MCS_Clonings' – Tabelle verwiesen wird, hier in einem View angezeigt.
- Primer_MCS_Mutations_View
    - Vergleichbar mit dem View ‚Primer_MCS_Clonings_View' werden hier die Daten der verwendeten Primer der ‚MCS_Mutations' – Tabelle zusammengefasst.
- Primer_Species_View
    - In diesem View werden zu den Primern die zugeordneten Spezies-Namen angezeigt.
- RestEnzymes_MCS_Clonings_View
    - Hier werden zu den Klonierungs-Informationen einer Multiple Cloning Site die Daten der verwendeten Restriktionsenzyme angezeigt.

## *3.3. Bedienungsanleitung*

Das Client-Programm benötigt keine Installations-Routine, sondern nur den Start der Anwendungsdatei.
Nach dem Start der Software erscheint das Login-Fenster, wo der Benutzer die IP-Adresse bzw. den Computernamen des Servers eingeben kann. Neben dieser Eingabe muss sich der Benutzer mit einem zugeteilten Benutzernamen und Kennwort authentifizieren. Bei der Eingabe des Benutzernamens spielt die Groß- und Kleinschreibung keine Rolle, jedoch bei der Eingabe des Kennworts schon.
Des Weiteren ist es möglich, zu überprüfen ob der benötigte SQL- und FTP Server erreicht werden kann.

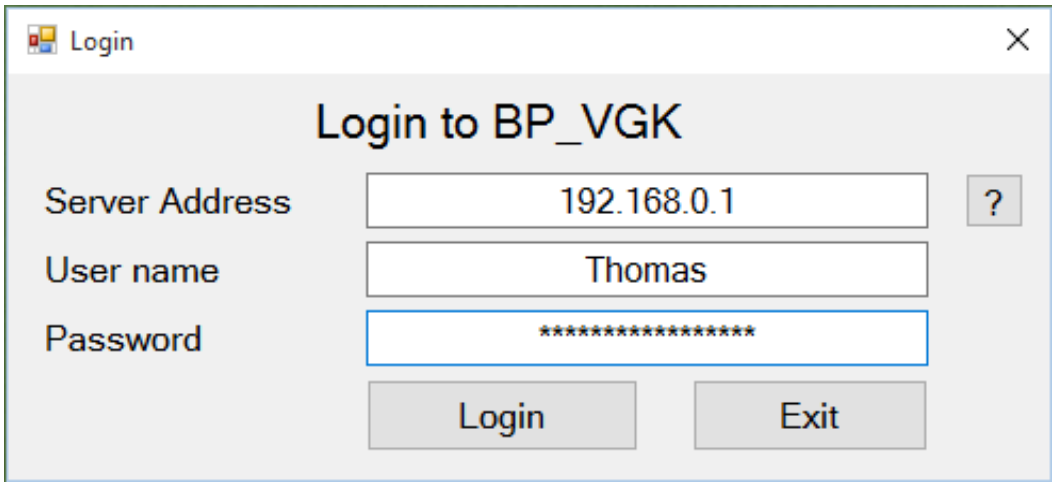



*Abbildung 8: Das Login-Fenster des Clients*

Nach dem erfolgreichen Anmeldevorgang erscheint das Hauptfenster des Clients. Dieses besitzt neben einer Menüleiste am oberen Fensterrand eine Statusleiste am unteren Fensterrand.

Folgende Inhalte hat die Menüleiste:

- In der Menüleiste unter dem Punkt ‚File' befindet sich der Punkt ‚Exit' um das Programm ordnungsgemäß zu schließen. Unter dem Punkt ‚Extras' befinden sich für alle Benutzergruppen der Punkt ‚Change own password' um das eigene Passwort zu ändern. Des Weiteren werden für Administratoren des Programms die Punkte ‚User administration' – zur Verwaltung der Benutzer-Accounts – sowie der Punkt ‚Manage tooltips' – zur Verwaltung von Pop-up-Fenstern – angezeigt.
- In der Statusleiste werden der aktuelle Benutzername des Benutzers, sowie der Name bzw. die IP-Adresse des verwendeten Servers angegeben.

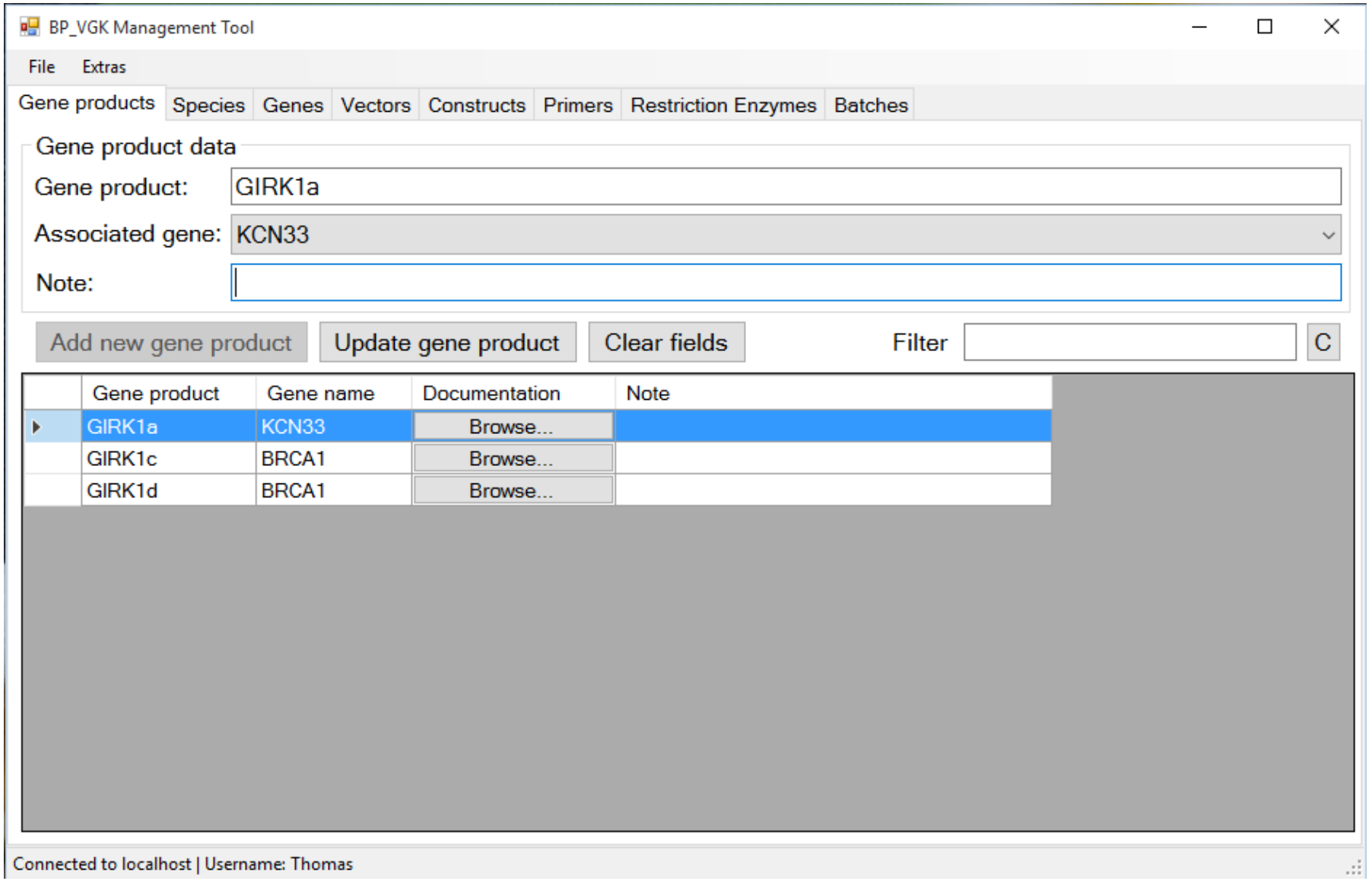



*Abbildung 9: Das Hauptfenster des Client-Programms mit geöffneten Tab zur Verwaltung der Gen-Produkte*

### 3.3.1. Benutzerverwaltung

Jeder Administrator hat die Möglichkeit Benutzer-Accounts zu erstellen, ändern oder zu löschen. Folgende Daten können bei einem Benutzer-Account von einem Administrator verändert werden.

- Benutzername
- Passwort - hier wird nicht das Passwort in Klartext-Form, sondern nur in einer Prüfsumme hinterlegt, um das Auslesen des richtigen Passworts zu verhindern.
- Activated - Aktiviert bzw. deaktiviert den Benutzer-Account.
- Administrator - Gibt an, ob der Benutzer Administratorrechte besitzt. Mit Administratorrechten kann man unter anderem Benutzer verwalten und Batches von anderen Benutzern übernehmen.
- Comment - Einfache Kommentarfunktion zu einem Benutzer-Account.

Des Weiteren kann man per Filter-Funktion nach Benutzern suchen.

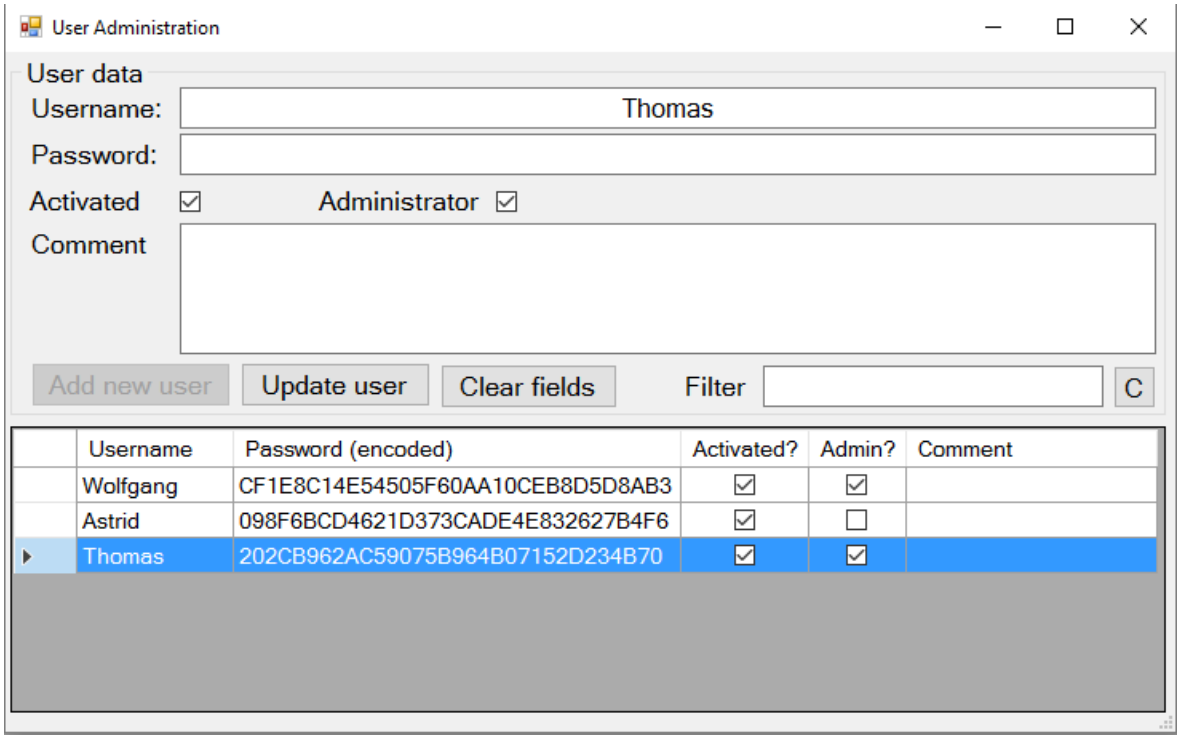



*Abbildung 10: Die Oberfläche zur Verwaltung der Benutzer-Accounts*

Die allgemeine Funktionalität ähnelt der eines Tabs im Hauptfenster (siehe 3.3.4.1). Bei der Aktualisierung eines Benutzers ist bei der Veränderung des Passwortes folgendes zu beachten: Wird kein Passwort in das dazugehörige Textfeld eingetragen, wird das Passwort des Benutzers nicht verändert. Somit kann kein

leeres Passwort einem Benutzer zugeordnet werden. Soll das Passwort verändert werden, muss wie bei den anderen Feldern das ausgewählte Passwort in das Textfeld eingetragen werden.

### 3.3.2. Änderung des eigenen Passworts

Jeder Benutzer kann, unabhängig seines Administrator-Status sein eigenes Passwort verändern. Dies geschieht im Hauptfenster über „Extras“ – „Change own password“.

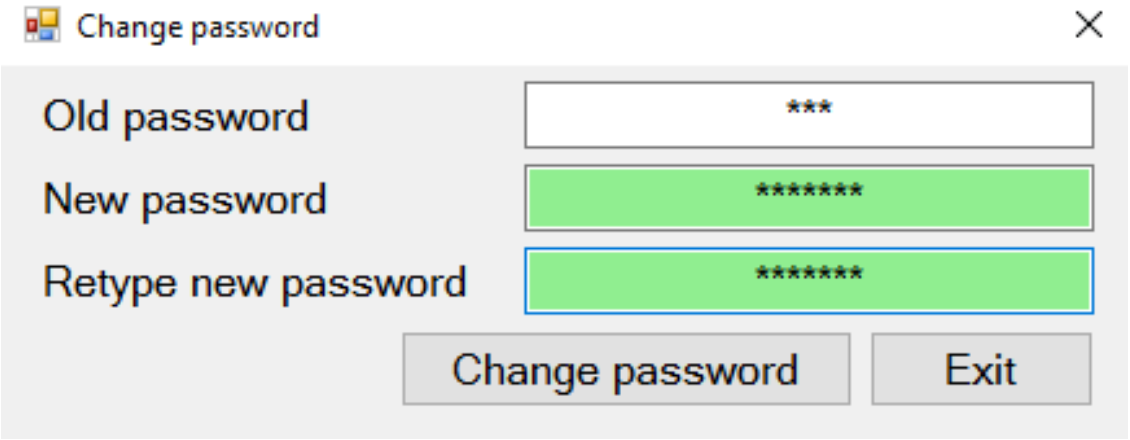



*Abbildung 11: Fenster zum Verändern des eigenen Passworts*

Nach der Eingabe des bestehenden Passworts kann das neu gewählte Passwort in die Textfelder eingetragen werden. Um allfällige Schreibfehler beim neuen Passwort vorzubeugen, muss dieses zweimal angegeben werden. Stimmen beide neuen Passwörter überein färben sich die beiden Textfelder grün.
Zur Änderung des Passworts genügt nun ein Klick auf den Button „Change password“.

### 3.3.3. Tool-Tips Verwaltung

Pop-Up Fenster werden über den Benutzersteuerelementen angezeigt, wenn einerseits der Benutzer den Mauszeiger über dem Element hält, andererseits ein Text zu diesem Element hinterlegt wurde.
Diese am SQL Server gespeicherten Einträge werden in einem Schlüssel-Wert-Paar abgespeichert. Aufgerufen kann die Verwaltung der Tool-Tips im Hauptfenster im Menü über „Extras“ – „Manage tooltips“. Diese Funktionalität steht nicht nur Benutzern mit dem Administrator-Privileg zur Verfügung, sondern allen registrierten Benutzern.

Der Schlüssel ist eine im Programm vorgegebene Zeichenfolge, die das Benutzersteuerelement kennzeichnet. Der Wert dieses Paares ist die Zeichenfolge, welche dann bei Bedarf über dem Element angezeigt wird.
Der Schlüssel der Einträge hat am Beispiel von ‚GENE-PRODUCT--TB_GENE-PRODUCT-NAME' folgenden einheitlichen Aufbau:

- GENE-PRODUCT: Der Prefix des Schlüssels beschreibt den Tab bzw. das Fenster in dem der Tool-Tip angezeigt wird. Nach disem Prefix folgen zwei Bindestriche „--„.
- TB: Diese Abkürzung, welche aus zwei bis drei Buchstaben besteht, bezeichnet die Art des Benutzersteuerelements, welchem dieser Tool-Tip zugeordnet ist. Folgende Abkürzungen werden verwendet:
    - TB (**T**ext**b**ox): Ein- oder mehrzeiliges Textfeld.
    - CB (**C**ombo**b**ox): Drop-Down Liste zur Auswahl eines Eintrages in einer vorgegebenen Liste.
    - BT (**B**u**t**ton)
    - DTP (**D**ate**T**ime**P**icker): Feld zur Auswahl eines Datums.
    - CLB (**C**hecked**L**ist**B**ox): Eine Liste mit mehreren Feldern, wobei ein oder mehrere Einträge ausgewählt werden können.
    - LV (**L**ist**V**iew): Dieses Element enthält eine Liste von Einträgen.

  Nach dieser Abkürzung folgt ein „_“ zur Abgrenzung zum dritten Teil des Schlüssels.
- Der dritte Teil des Schlüssels beschreibt das Benutzersteuerelement an sich.

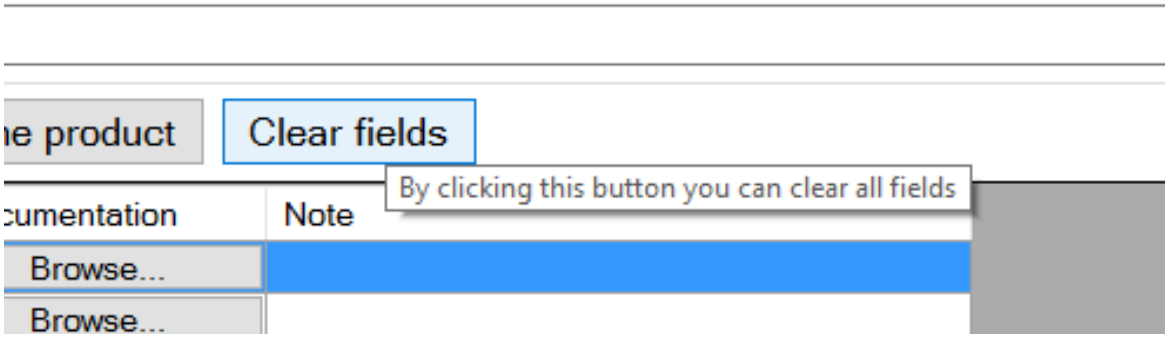



*Abbildung 12: Ein Pop-up Fenster erscheint wenn der Mauszeiger über dem Benutzersteuerelement gehalten wird*

Die Verwaltungsoberfläche besteht aus einer zweispaltigen Tabelle. Die linke Spalte enthält den Schlüssel zu dem Tool-Tip, die rechte den Text, der angezeigt werden soll.

Die Speicherung der Tool-Tips erfolgt beim Schließen des Fensters. Aktualisiert werden die Tool-Tips nach einem Neustart der Software.

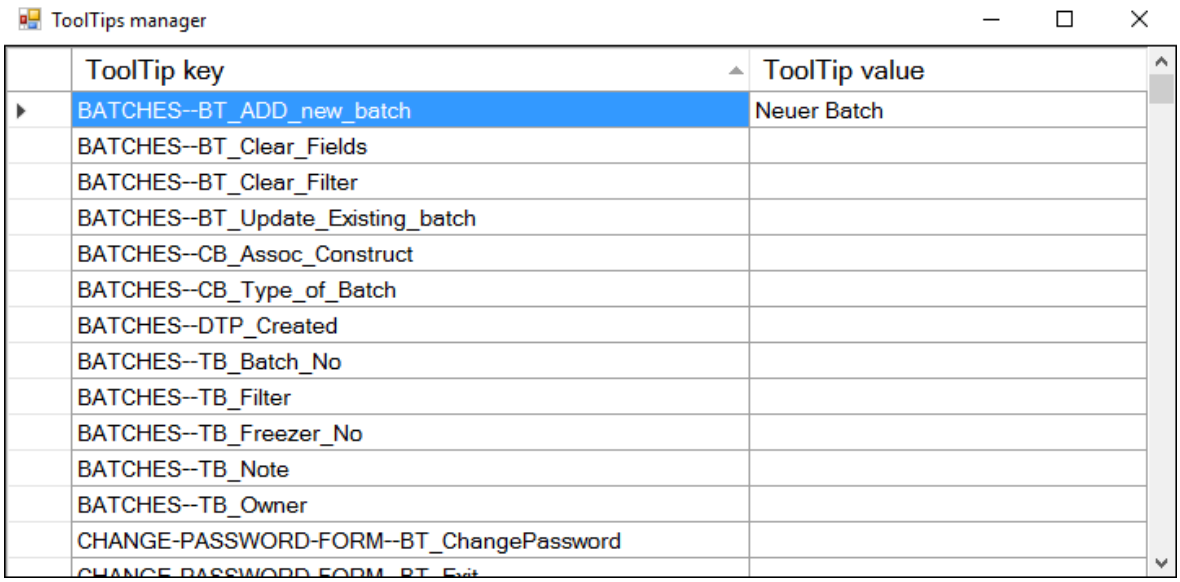



*Abbildung 13: Fenster zur Verwaltung der Tooltips*

### 3.3.4. Beschreibung der einzelnen Tabs

In den folgenden Unterkapiteln werden die acht aufrufbaren Tabs der Client-Software beschrieben. Alle, außer das Tab zur Verwaltung der Konstrukte, haben vom Prinzip her den gleichen Aufbau. Auf der oberen Hälfte des Fensters befindet sich die Eingabe-Maske, darunter drei Buttons und am unteren Ende befindet sich eine Tabelle zur Anzeige der gespeicherten Datensätze.

Alle Daten, die in dieser Arbeit in den Tabs angezeigt werden, sind Testdaten und sind nicht für einen produktiven Einsatz gedacht.

Die Abhängigkeiten zwischen den Tabs veranschaulicht die Grafik „Abbildung 14: Grafische Darstellung der Abhängigkeiten der Tabs untereinander“. In dieser Grafik sieht man, dass sich die Inhalte aller Tabs im Tab „Constructs“ sammeln. Des Weiteren befinden sich die Daten des Tabs „Constructs“ im Tab „Batches“ und umgekehrt.

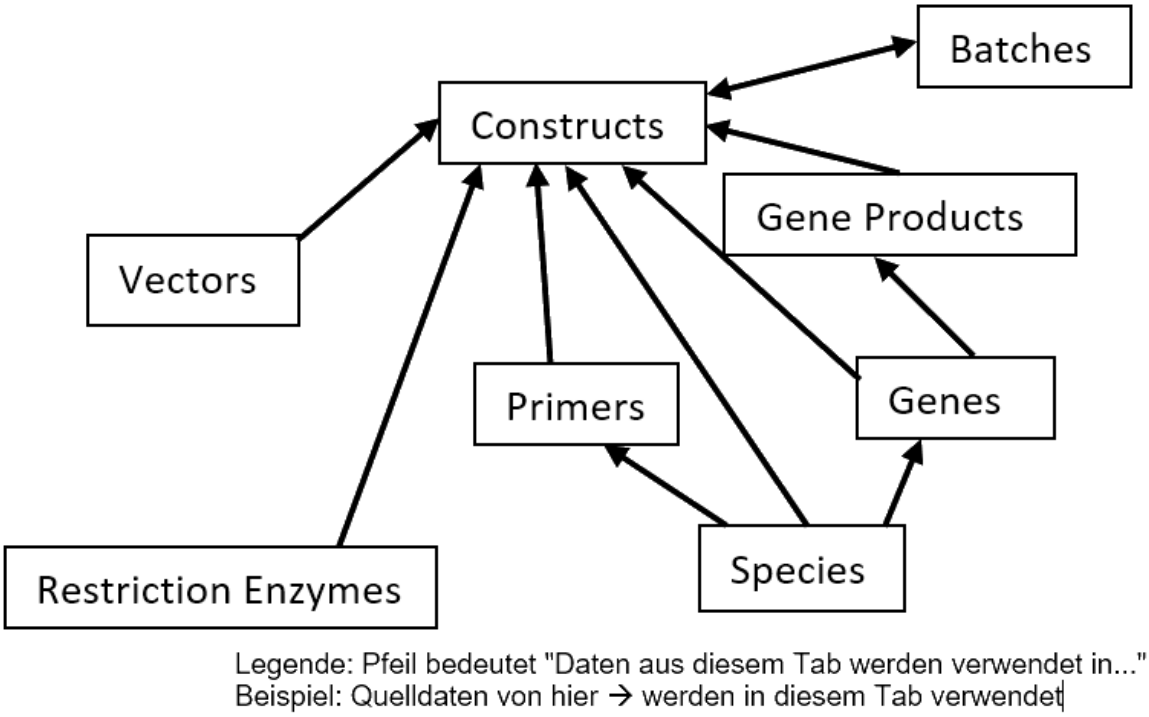



*Abbildung 14: Grafische Darstellung der Abhängigkeiten der Tabs untereinander*

#### 3.3.4.1. Prinzipieller Aufbau eines Tabs

Da der Tab zur Verwaltung der Spezien den allgemeinen Aufbau am besten veranschaulicht, wird der prinzipielle Aufbau eines Tabs an diesem hier beschrieben.

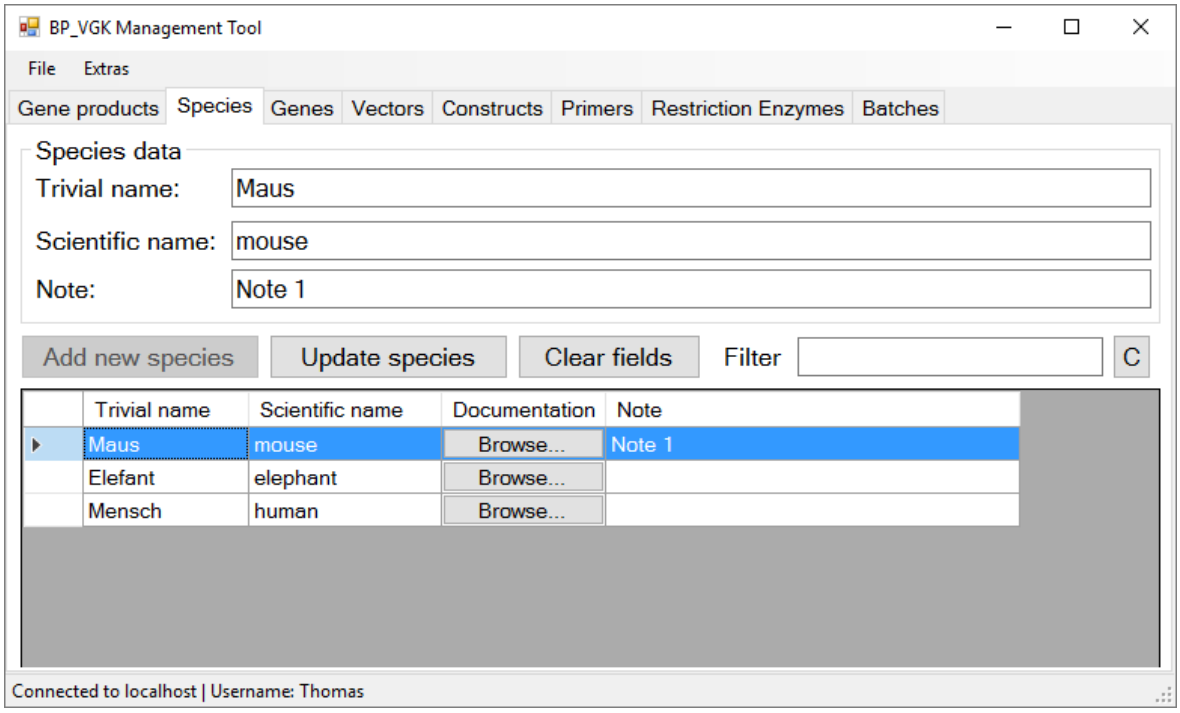



*Abbildung 15: Ein typischer Tab der Client-Software*

Der Eingabebereich befindet sich in der umrahmten Zone im oberen Bereich des Tab. Darunter befinden sich zwei Buttons zum Hinzufügen bzw. Aktualisieren eines Datensatzes. Mit dem Button ‚Clear fields' lassen sich die Felder im Eingabebereich leeren.
Beim Starten der Software befindet sich der Eingabebereich im ‚Neuen Datensatz'-Modus. Sobald eine Eingabe in die Felder durchgeführt wurde, wird der ‚Add new' Button freigeschalten. Somit lässt sich der Datensatz in die Datenbank einfügen. Um einen Datensatz zu aktualisieren, muss dieser in der Tabelle ausgewählt werden. Nach der Auswahl werden die Felder im Eingabebereich mit den Daten des Datensatzes aufgefüllt und der ‚Update'-Button zum Aktualisieren des Datensatzes wird freigeschalten.
Damit man in den ‚Neuen Datensatz'-Modus zurückkommt bzw. den Aktualisierungsvorgang für einen bestehenden Datensatz unterbricht, muss der ‚Clear fields'-Button gedrückt werden. Dieser bricht den ‚Update'-Modus ab und leert alle Eingabefelder.
Rechts neben dem Button ‚Clear fields' befindet sich die Möglichkeit zum Filtern der Datensätze. Hier kann ein beliebiger Suchbegriff eingegeben werden. Sofort nach Eingabe des Suchbegriffs werden die Tabellen Einträge abgesucht, die den Suchbegriff beinhalten. Wenn Übereinstimmungen von Datensätzen mit diesem Suchbegriff gefunden werden, werden nur mehr diese Datensätze in der Tabelle aufgelistet. Falls keine Übereinstimmung gefunden wird, erscheint die Tabelle ohne Einträge. Um die Filterfunktion wieder zu deaktivieren kann entweder das Textfeld geleert werden bzw. mit dem Klick auf den ‚C'-Button rechts des Textfelds dies in Auftrag gegeben werden.
Um einen Datensatz zu löschen muss auf das Tabellenfeld links des Datensatzes mit der rechten Maustaste geklickt werden. Daraufhin erscheint ein Pop-Up (siehe Abbildung 16: Bestätigung des Löschvorgangs für einen Datensatz) bei der Position des Mauszeigers ein Pop-Up mit der Frage, ob dieser Datensatz wirklich gelöscht werden soll. Ist dies der Fall, kann dies mit einem Klick auf die Frage bestätigt werden. Beim Löschvorgang eines Datensatzes ist darauf zu achten, dass auch die hinterlegte Dokumentation zu diesem Datensatz mitgelöscht wird.

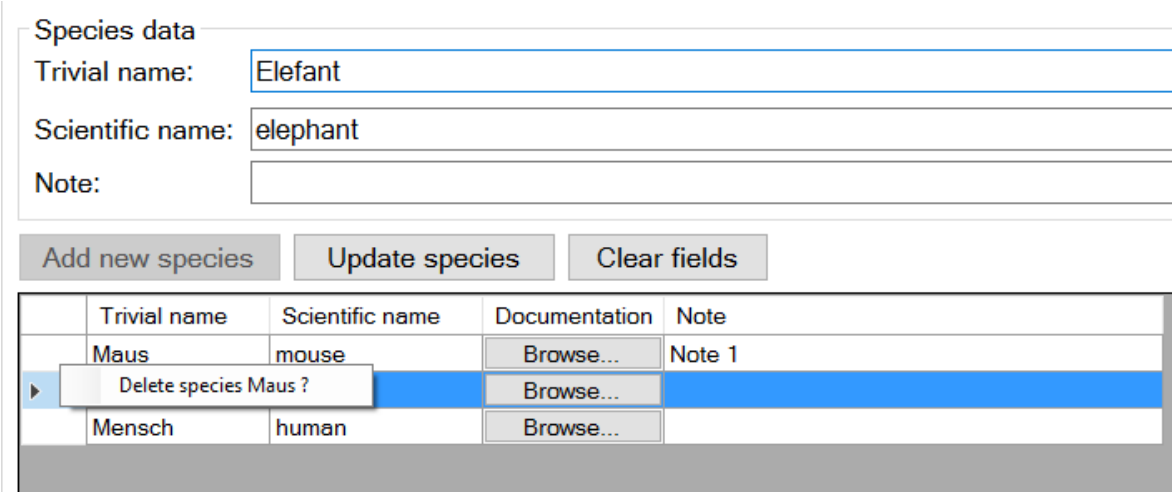



*Abbildung 16: Bestätigung des Löschvorgangs für einen Datensatz*

Die Tabelle, welche die bestehenden Datensätze beinhaltet, kann je nach Bedarf nach einzelnen Spalten sortiert werden. Nicht nur die eingangs erwähnte Löschfunktion eines Datensatzes befindet sich in jeder Tabelle, sondern auch die Informationen über die jeweiligen Datensätze. Zusätzlich werden folgende Funktionen ermöglicht:

- In der vorletzten Spalte befindet sich der Button zum Aufrufen der Dokumentation zu dem jeweiligen Datensatz.
- In der letzten Spalte befindet sich die Notizspalte für allfällige Bemerkungen zu dem jeweiligen Datensatz.

#### 3.3.4.2. Verwaltung der Dokumentation zu einem Datensatz

Hinter jedem Datensatz, welcher vom Benutzer eingegeben werden kann, können verschiedene Dateien und Ordner abgelegt werden. In dieser Verwaltungsoberfläche wird am oberen Ende des Fensters angezeigt, zu welchem Datensatz Dokumente und Ordner hinterlegt werden.

In der Baumansicht darunter wird angezeigt, welche Dokumente und Ordner für diesen Datensatz bereits hinterlegt sind. Diese Elemente werden im Hintergrund auf einem FTP-Server abgelegt.

**Ordner erstellen**

Um einen Ordner zu erstellen gibt man den gewünschten Ordnernamen in dem Textfeld links des Buttons „Create directory“ ein und klickt auf den Button. Sogleich erscheint der Ordner in der Baumansicht. Hierbei ist zu beachten, dass keine Unterordner erstellt werden können.

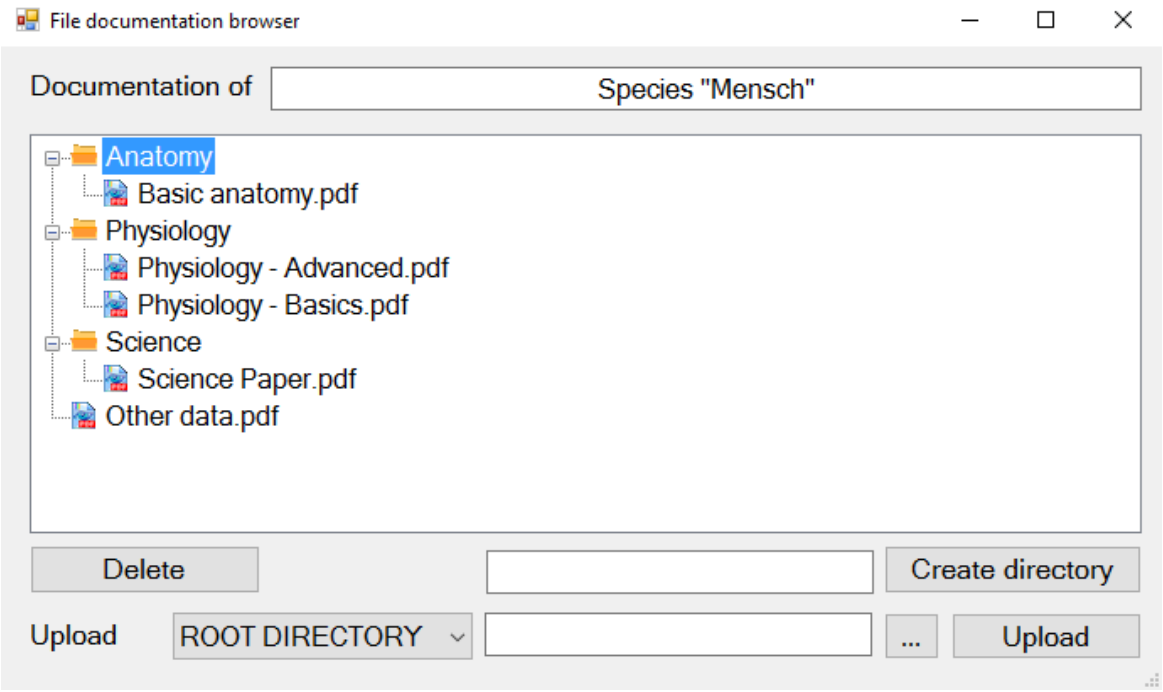



*Abbildung 17: Fenster zur Verwaltung der Dokumentation zu einem Datensatz*

**Dateien hochladen**

Um Dateien hochzuladen wählt man zuerst den Ordner, in den die Datei hochgeladen werden kann. Dies geschieht mithilfe des Auswahlfeldes rechts neben dem Feld „Upload“. Der erste Eintrag in diesem Auswahlfeld ist immer „ROOT DIRECTORY“ – mithilfe diesem Eintrag lässt sich die Datei in das Hauptverzeichnis der Dokumentationsverwaltung hochladen. Sobald weitere Ordner in der Verwaltungsoberfläche angelegt wurden, können diese dann auch hier ausgewählt werden. Nach der Auswahl des Ordners wird über den Button mit Beschriftung „…“ ein Dialogfenster geöffnet und die Datei ausgewählt. Der Pfad der Datei erscheint daraufhin im Textfeld. Nach einem Klick auf den „Upload“-Button wird die Datei hochgeladen und in der Baumansicht angezeigt.

**Löschen von Dateien und Ordner**

Um Dateien und Ordner zu löschen wird das zu löschende Element in der Baumansicht markiert. Nach einem Klick auf den „Delete“-Button ist das Element gelöscht. Falls es sich bei dem Element um einen nicht leeren Ordner handelt erfolgt eine Sicherheitsabfrage, ob der Ordner inklusive seiner Dateien wirklich gelöscht werden sollen.

**Aufrufen einer Datei**

Zum Aufrufen einer Datei in der Dokumentation, muss die Datei mit einem Doppelklick aufgerufen werden. Ist für die Dateiendung am Computer eine

Anwendung hinterlegt (z.B. Adobe Acrobat Reader für die Dateiendung „.pdf“) wird die Datei sofort mit diesem Programm geöffnet. Falls dem nicht so ist, wird ein Dialog zur Abfrage des Speicherortes für die Datei angezeigt. Somit kann die Datei auf jedem beliebigen Speicherort abgelegt werden.

#### 3.3.4.3. Tab - „Gene“

In diesem Tab können Gene angelegt, verändert und gelöscht werden. Außerdem können diese verschiedenen Spezies zugeordnet werden. Dies geschieht über die Auswahlbox rechts neben den Eingabefeldern. Des Weiteren kann auch ein mehrzeiliger Kommentar bzw. eine Dokumentation zu jedem Gen hinterlegt werden. Außerdem ist zu beachten, dass zu jedem Gen nur eine Notiz hinterlegt werden kann. Somit ist es nicht möglich, dass zu jeder Gen-Spezies-Kombination eine eigene Notiz hinterlegt werden kann.
Bei der allgemeinen Dokumentation, bei welcher Dateien und Ordner hinterlegt werden können, ist wiederum möglich für jede Kombination eine eigene Dokumentation zu hinterlegen. Gibt es keine Zuordnung zu einer Spezies, ist eine Hinterlegung einer Dokumentation nicht möglich.

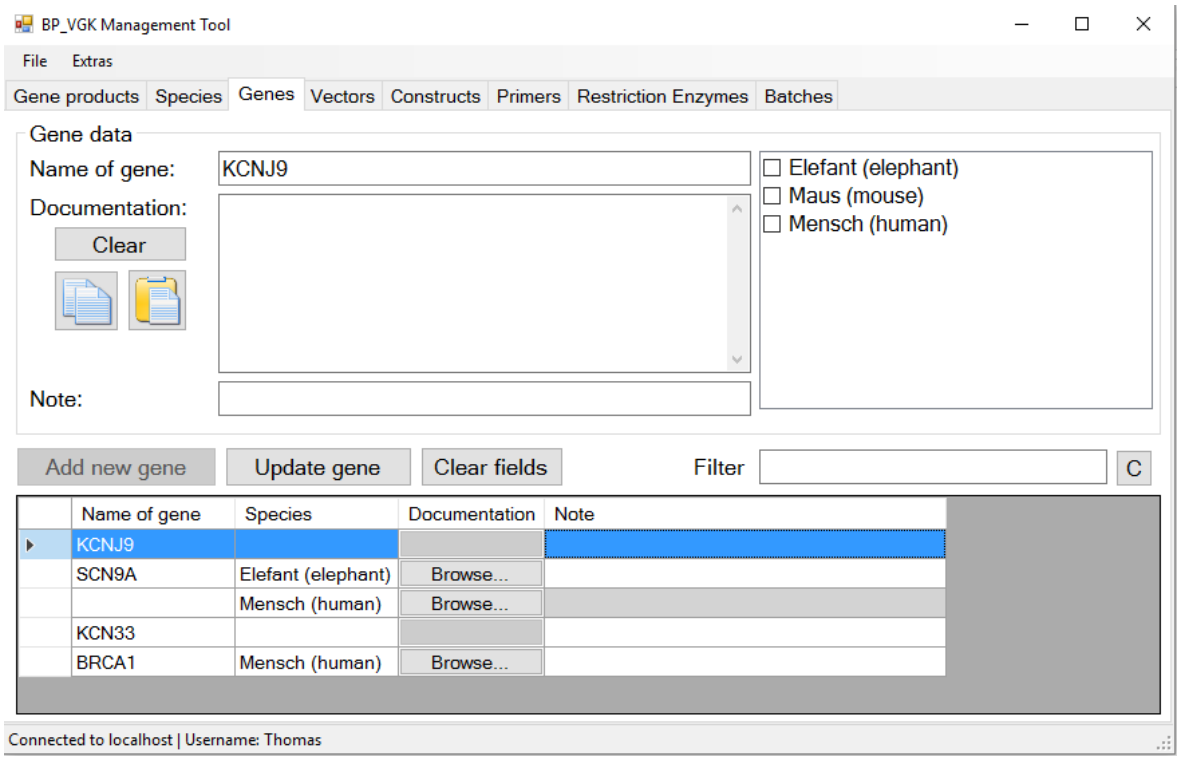



*Abbildung 18: Tab zur Verwaltung der Gene*

### 3.3.4.4. Tab - „Primers“

Neben dem Namen des Primers kann in diesem Tab die Vorwärts- als auch Rückwärts-Sequenz angegeben werden. Zusätzlich ist es möglich den Primer einer gewissen Spezies zuzuweisen. Die weiteren Eingabefelder wie das Notiz-Feld oder die Filter-Funktion verhalten sich analog zu den anderen Tabs.
Rechts neben diesen Eingabefeldern kann die Art des Primers angegeben werden. Dies ist besonders wichtig, da aufgrund dieser Auswahl die Primer-Listen beim Zusammenstellen der gentechnischen Konstrukte vorgegeben werden.

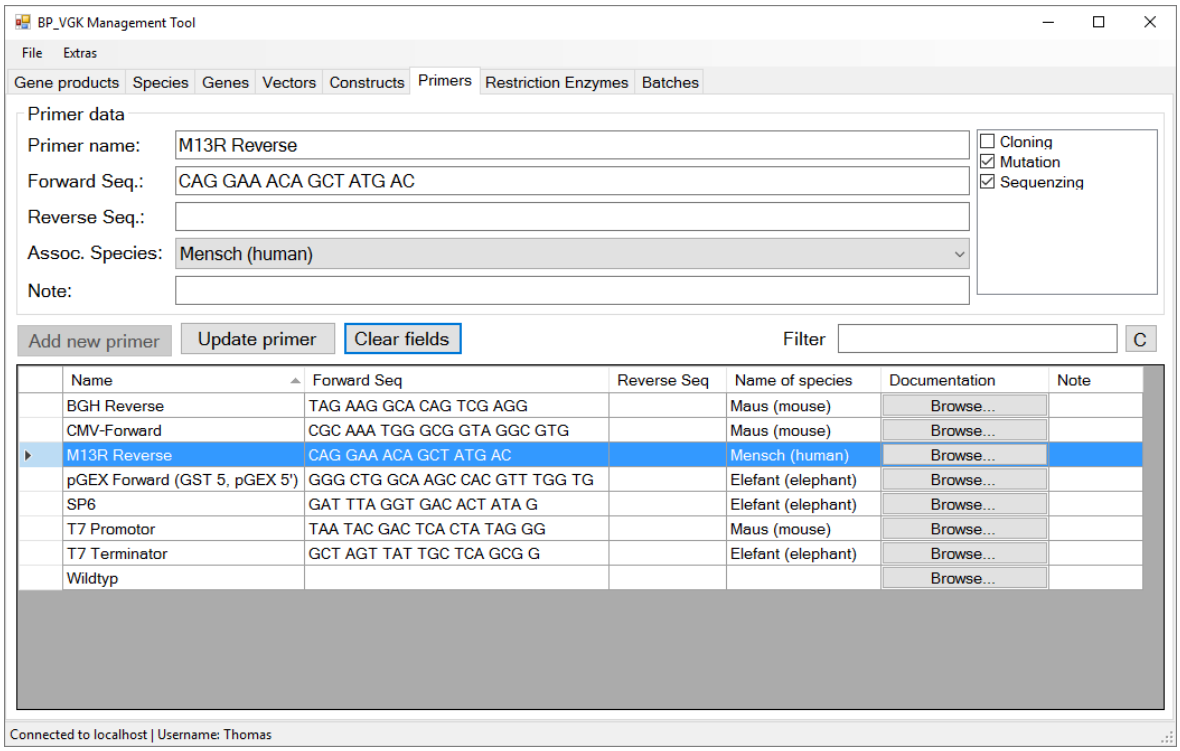



*Abbildung 19: Tab zur Verwaltung der Primer*

### 3.3.4.5. Tab - „Restriction Enzymes“

Der Tab zur Verwaltung der Restriktionsenzyme enthält neben den beiden Textfeldern zur Verwaltung von Name und einer Notiz zu einem Restriktionsenzym keine weiteren Felder.

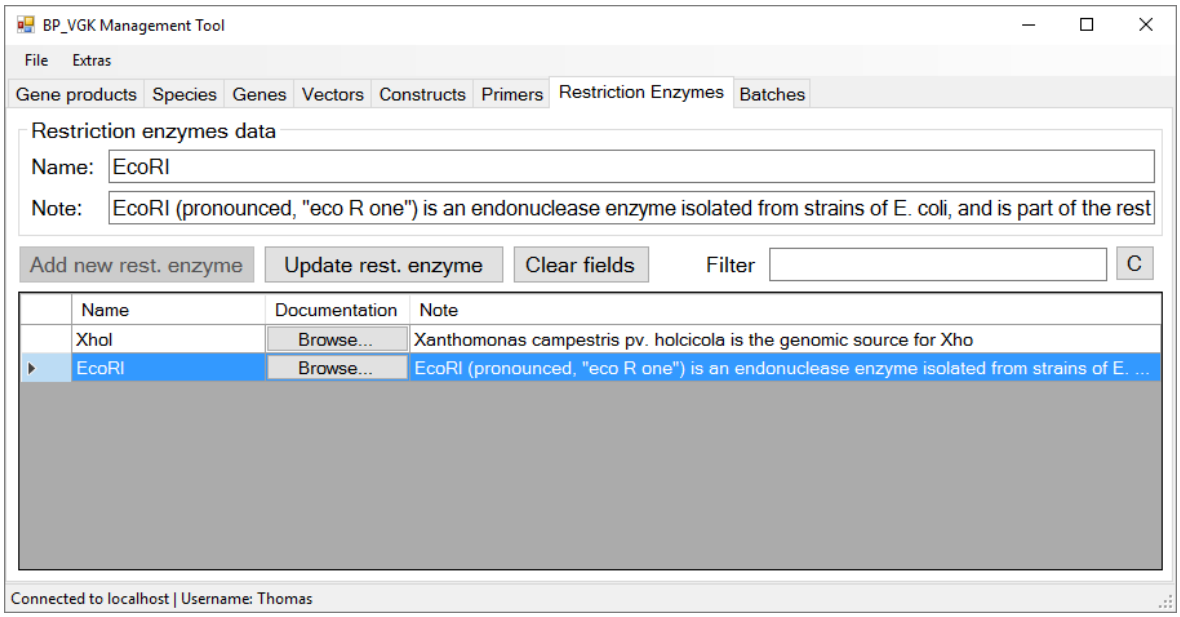



*Abbildung 20: Tab zur Verwaltung der Restriktionsenzyme*

### 3.3.4.6. Tab - „Batches“

Jedem Batch kann neben einer „Freezer Nr.“ auch ein Konstrukt zugeordnet werden. Somit ist es möglich, dass bei jedem Konstrukt die zugehörigen Batches angezeigt werden.

Des Weiteren können folgende Eigenschaften zu einem Batch zugeordnet werden:

- Typ des Batches (DNA oder RNA)
- Datum der Erstellung des Batches
- Besitzer des Batches

Jeder Benutzer sieht jedoch nur folgende Batches:

- Alle Batches, denen kein Besitzer zugeordnet ist
- Alle Batches, wenn er selbst Administrator-Rechte hat
- Alle Batches, die ihm zugeordnet sind.

Darüber hinaus kann jeder Administrator von anderen Benutzern Batches übernehmen. Hierzu muss ein Batch eines anderen Benutzers aufgerufen werden und in das Textfeld, wo der Name des Besitzers angezeigt wird, ein Doppelklick gemacht werden. Nach positiver Antwort auf die Frage, ob der Batch übernommen werden soll, muss der Batch mittels Klick auf den Button „Update batch“ noch aktualisiert werden.

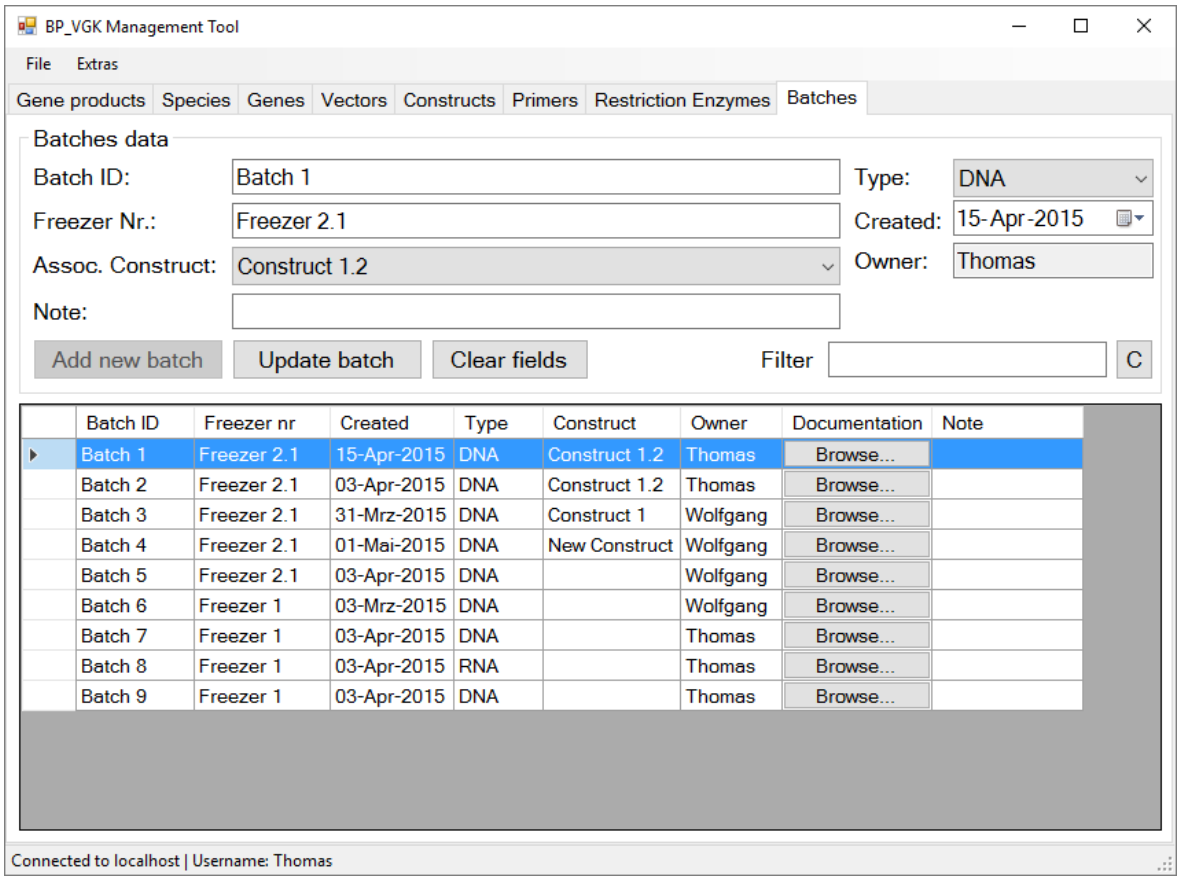



*Abbildung 21: Tab zur Verwaltung der Batches*

### 3.3.4.7. Tab - „Constructs“

Im Gegensatz zu den restlichen Tabs in der Client-Software konnte in diesem Tab das allgemeine Design nicht eingehalten werden. So musste die Tabelle, in welcher die bestehenden Datensätze stehen, auf den rechten Bildschirmrand weichen.
Folgende Daten können zu einem Konstrukt hinterlegt werden:

- Name des Konstrukts
- Verwendeter Vektor
- Datum der Erstellung und Aktualisierung des Datensatzes
- Name des Benutzers, der den Datensatz erstellt bzw. aktualisiert hat
- Sequenzierungsdaten (Verwendeter Primer inkl. Freitext)
    - Hier können beliebig viele weitere Sequenzierungsdaten angegeben werden. Mit dem Button „+“ werden weitere Zeilen hinzugefügt; bei Klick auf den „-“ - Button wird die jeweils letzte Zeile der Sequenzierungsdaten entfernt.
- Linearisierungsdaten (Verwendetes Restriktionsenzym inkl. Freitext)
- Notiz

Des Weiteren werden in der rechten oberen Ecke die zugeordneten Batches angezeigt. Mit einem Doppelklick auf einen Batch wird dieser im Batches-Tab angezeigt. Falls der verwendete Benutzer kein Administrator ist und weitere Batches, die anderen Benutzern zugeordnet sind, vorhanden sind, wird angezeigt, dass noch weitere Batches verfügbar wären.
In dem Bereich, wo die zugehörigen Batches angezeigt werden, sind diese wie folgt formatiert:

*{Typ des Batches (DNA oder RNA)} {Name des Batches} ( {Besitzer des Batches} / {Datum der Erstellung des Batches}*

Weiters kann ein Konstrukt dupliziert werden. Hierbei werden alle hinterlegten Daten zu einem Konstrukt in ein neues Konstrukt kopiert. Damit es zu keinen Vertauschungen kommen kann, wird beim Namen des neuen Konstrukts „ (copy)“ hinzugefügt. Auch die Dokumentation wird in das neue Konstrukt kopiert. Dies geschieht im Hintergrund und kann je nach Größe und Menge an Dokumentation eine gewisse Zeit in Anspruch nehmen.

#### 3.3.4.8. **Multiple Cloning Site**

Jedem Konstrukt können mehrere Multiple Cloning Sites zugeordnet werden. Hinzugefügt wird mit dem Klick auf „Add MCS“, gelöscht wird es durch den „Remove“ – Button in der MCS.
Zuerst muss das verwendete Gen ausgewählt werden. Daraufhin werden die Auswahlfelder zur Spezies und zum Gen-Produkt aufgefüllt. Je nachdem welche Spezies dem Gen zugeordnet ist, wird dann angezeigt. Das gleiche gilt für das Auswahlfeld zum verwendeten Gen-Produkt. Rechts neben dem Auswahlfeld befindet sich ein „Browse“-Button zum Aufruf der Dokumentation des verwendeten Gen-Produkts.
Für die Information über die ausgewählte Mutation steht einerseits ein Textfeld als Freitext, andererseits einer Liste aller Primer, welche als Mutations-Primer definiert worden ist, zur Verfügung. Auch hier ist es wieder möglich mehrere Mutationen hinzuzufügen. Wiederum kann mithilfe des „+“-Buttons eine weitere Mutation eingefügt werden, mit dem „-‘‘-Button die unterste Mutation entfernt werden.
Pro Klonierungs-Information, welche wie die Mutationen hinzugefügt bzw. wieder entfernt werden können, kann jeweils ein Restriktionsenzym und ein Primer (welchem die Klonierungs-Eigenschaft zugewiesen wurde) hinterlegt werden.

Nach der Erstellung einer Multiple Cloning Site muss diese noch per Klick auf den „Save“-Button gespeichert werden.

#### 3.3.4.9. **Weitere Tabs**

**Gene Products**

In diesem Tab können Gen-Produkte angelegt und jeweils einem bestimmten Gen zugeordnet werden.

**Vectors**

Neben dem Namen des Vektors kann eine mehrzeilige Text-Dokumentation, wie bei der Verwaltung der Gene (siehe Kapitel 3.3.4.3), hinterlegt werden.

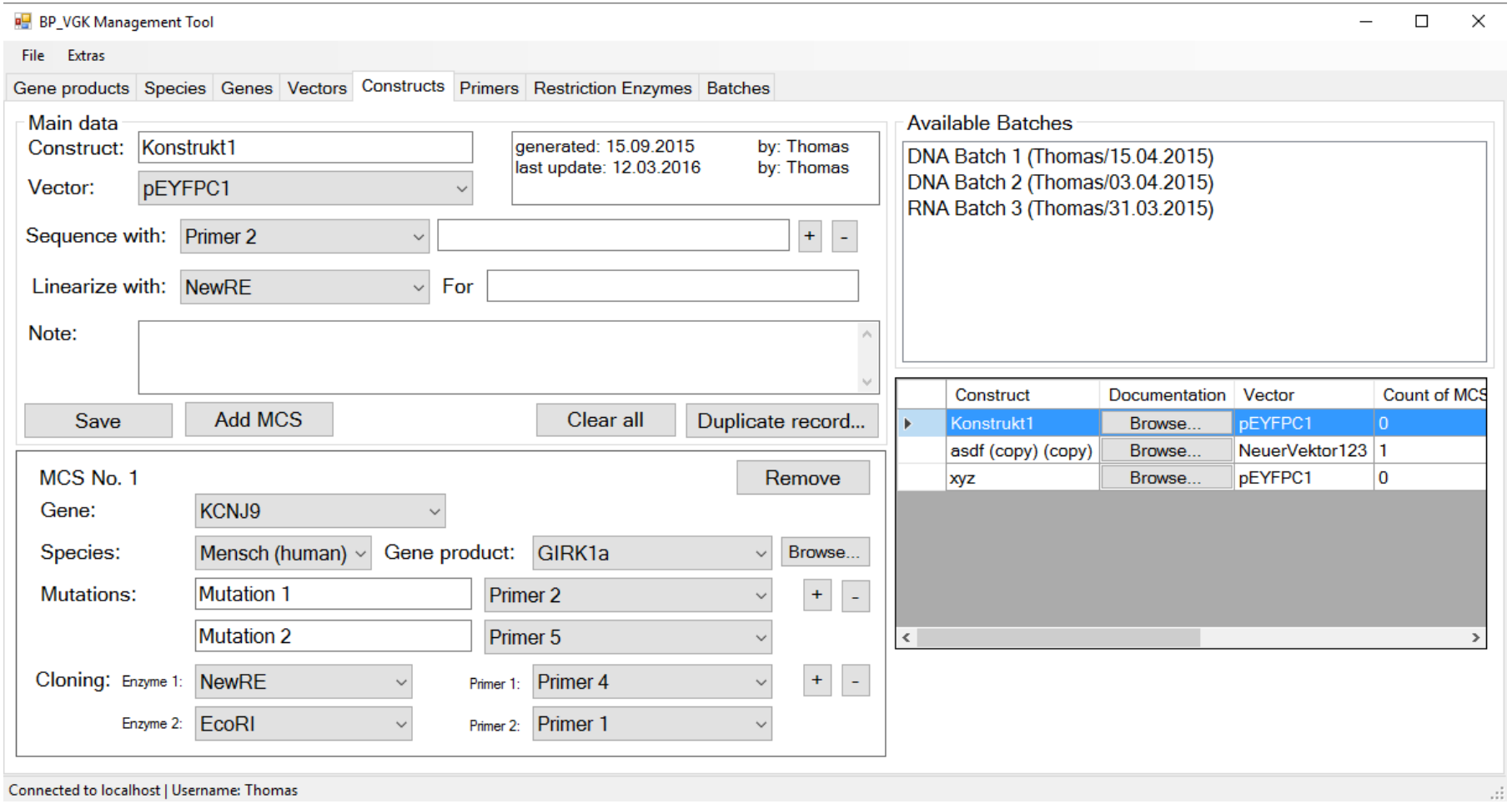



*Abbildung 22: Tab zur Verwaltung der gentechnischen Konstrukte*

## *3.4. „Programmers Manual“*

Dieses Kapitel soll helfen, bei einer zukünftigen Erweiterung der Software einen einheitlichen Programmier-Stil gewährleisten zu können. Der Quellcode der Software liegt dieser Diplomarbeit bei und kann mithilfe der Entwicklungsumgebung Microsoft Visual Studio 2013 Professional (oder kompatibler Software) selbst verändert bzw. neu kompiliert werden.
Im restlichen Teil des Kapitels werden Kenntnisse in der Software-Entwicklung benötigt.
Des Weiteren befindet sich der Quellcode des Tabs „Batches“ im Anhang dieser Arbeit (siehe Kapitel 6.1). Dieser Tab wurde mit Kommentaren versehen um die Vorgehensweise im Quellcode zu erklären.

### 3.4.1. Hinzufügen eines neuen Tabs

In der Datei „MainForm.cs“ (Projekt „BP_VGK.VerwaltungGentechKonstrukte“) kann ein neuer Tab hinzugefügt werden. Der Inhalt eines Tabs wird mithilfe von UserControls (Ordner „UserControls“) realisiert. Im neu angelegten UserControl wird dann Property „*Dock*“ auf „*Fill*“ gesetzt.

### 3.4.2. Aufbau eines neuen Tabs

Nach folgendem Stil sind alle Tabs der Client-Software aufgebaut. Dies hat zur Folge, dass beim Einbau eines neuen Tabs der Programmier-Aufwand reduziert werden kann:

- In der Oberfläche wird im oberen Bereich eine *GroupBox* erstellt in welcher die Eingabefelder und die dazugehörigen Buttons abgelegt werden. Damit die GroupBox sich an die Größe des Fensters anpasst wird daraufhin die *Anchor*-Eigenschaft auf „Top, Left, Right“ gesetzt.
- Das *DataGridView* im unteren Bereich des Fensters wird mithilfe eines *DataSet* befüllt. Bei der Tabelle wird die *Anchor*-Eigenschaft auf „Top, Bottom, Left, Right“ gesetzt.
    - Hierbei ist darauf zu achten, dass nur ganze Zeilen markiert werden können.
    - Weiters soll eine Bearbeitung des Datensatzes nur über die Eingabefelder erfolgen können.

- Im Konstruktor werden die Tooltips zu den einzelnen Controls zugewiesen. Dies geschieht mit der Methode *InternalTools.SetToolTip(Control, string)*. Falls der Schlüssel zu dem Control noch nicht in der Datenbank ist, wird dieser Eintrag dort angelegt.
- Es werden Eventhandler an die verwendeten Tabellen angehängt, um über allfällige Änderungen darin informiert werden zu können. Danach werden diese ausgelöst, um die Listen und Tabellen mit den aktuellen Daten zu befüllen.
    - In der Methode muss vor dem Neuauffüllen der Liste o.ä. die ID des aktuellen Datensatzes in einer globalen Variable hinterlegt werden. Nach der Aktualisierung kann der vorher gewählte Datensatz somit wieder ausgewählt werden. Damit unterbricht eine Aktualisierung der Daten nicht die Arbeit des Benutzers.
- Zur Öffnung der Datei-Dokumentation im *DataGridView* wird ein Handler an das *CellClick*-Event angehängt. In diesem wird zuerst überprüft, ob auf ein Feld in der Datei-Dokumentations-Spalte geklickt wurde. Daraufhin wird die GUID des Datensatzes mithilfe der *Tools.GetIDOfSelectedRow (ref DatagridView, int indexOfIDColumn)* ausgelesen. Danach wird ein neues Objekt der Klasse *FileDocumentationBrowserForm* instanziert und neben der ID des Datensatzes der gewünschte Text im Textfeld übergeben.
  Die Verwaltung der Dokumentation übernimmt die *FileDocumentationBrowserForm*-Klasse selbst.
- Wenn ein Rechts-Klick auf den Zeilen-Kopf gemacht wird, soll ein *ToolStripMenuItem* erscheinen, welches nachfragt, ob dieser Datensatz gelöscht werden soll.

Natürlich können nicht alle Konventionen in einem UserControl erklärt werden. Daher wurde das UserControl zur Verwaltung der Batches im Sourcecode mithilfe von Kommentaren genau beschrieben.

### 3.4.3. Allgemeine Klassen

Alle allgemeinen Klassen, welche zur Objekt-Instanziierung verwendet werden, sollen im „BP_VGK.Common“-Projekt abgelegt werden. Auf dieses Projekt haben der FTP-, der SQL-Connector sowie die Benutzeroberfläche Zugriff.
Des Weiteren befinden sich im „BP_VGK.Common“-Projekt folgende Klassen:

- ConnectionsDataManager
    - GetMSSQLConnectionString() - liefert die Verbindungszeichenfolge für die Datenbankanbindung
    - Zugangsdaten für den FTP-Server
    - Zugangsdaten für den MSSQL-Server
- ToolTip_Constants
    - In dieser Klasse werden die statischen Namen für die Tool-Tips hinterlegt. In dieser ‚Haupt'-Klasse gibt es für jeden Tab bzw. jedes Fenster eine eigene Klasse mit den spezifischen, statischen Namen. Die Verwendung der statischen Namen wird im Abschnitt 3.4.2 erklärt.
- Allfällige Enumerationen werden in der Datei „Enums.cs" verwaltet
- IDItem
    - Jedes zu speichernde Objekt soll von diesem Objekt erben, damit es eine GUID als eindeutige Kennzeichnung besitzt.
- INISettings
    - Mit dieser Klasse kann auf die Settings.ini - Datei zugegriffen werden.
- Tools
    - Diese Klasse beinhaltet allgemeine Methoden, welche von den Tabs in der Client-Software benötigt werden.

### 3.4.4. Anbindung an den MSSQL Server

Jede Änderung an der Software hat zumeist zur Folge, dass eine Änderung in der Datenbank nötig ist. Damit eine Änderung an den MSSQL Server geordnet übergeben werden kann, muss ein SQL-Skript angelegt werden. Dieses kann in den Ordner „DB Version Upgrades" im Projekt „BP_VGK.Data.MSSQL" abgelegt werden. Nach dem Hinzufügen als Ressource zum Projekt kann im Konstruktor der DataProvider-Klasse ein Versions-Update kodiert werden. Dies geschieht am Beispiel für die Version 19 folgendermaßen:

```
if (dbVersion < 19) {
       sendTextToSQLServer(BP_VGK.Data.MSSQL.Properties.Resources.version19);
       setDBVersionNumber(19);
}
```

Falls eine neue Tabelle mit diesem Versions-Update erstellt wurde, kann ein SqlWatcher installiert werden. Dieser informiert über Änderungen in einer Tabelle. Dazu muss diesem SqlWatcher ein vollständiges SQL Select-Statement übergeben

werden. Das Event, welches vom SqlWatcher ausgelöst wird, kann dann auf ein selbst programmiertes Event umgeleitet werden, welches von den betreffenden Tabs empfangen werde kann.
Weiters werden folgende Methoden bei der Installation einer neuen Tabelle in der DataProvider-Klasse benötigt:

- AddOrUpdate: Dieser Methode wird ein Objekt der neuen Daten mitgegeben. Falls die ID des Objekts leer ist, wird das Objekt in die Datenbank eingefügt. Ansonsten wird der betreffende Datensatz aktualisiert. Der Rückgabewert der Methode ist:
    - Bei erfolgreicher Aktualisierung: Die GUID des aktualisierten Datensatzes
    - Wenn keine Zeile in der Datenbank verändert wurde: Eine leere GUID
    - Bei einem Einfüge-Vorgang: Die neu generierte GUID des neuen Datensatzes
- Delete: Dieser Methode wird die GUID des zu löschenden Datensatzes mitgegeben. Falls ein Datensatz gelöscht wird, wird dies über den SqlWatcher mitgeteilt.
- Get: Je nach Anforderung und Programmierung wird hier ein Array aus Objekten des neuen Typs zurückgegeben.

Beim Verfassen der SQL-Statements ist darauf zu achten, dass keine SQL-Injection möglich ist. Dies ist auf verschiedene Arten möglich, wobei hier die Parametrisierung des SqlCommands verwendet wurde. (19)

## *3.5. Präsentation der Diplomarbeit*

Im Rahmen eines wissenschaftlichen Jour-Fixe des Instituts für Biophysik am 21. Oktober 2015 wurde der damalige aktuelle Stand der Software den wissenschaftlichen Mitarbeitern vorgestellt. Im Rahmen dieser Präsentation wurden weitere Ideen gesammelt um die Software noch benutzerfreundlicher und intuitiver zu machen.

# 4. Diskussion

Es wurde in Zusammenarbeit mit den Mitarbeitern des Instituts für Biophysik der Medizinischen Universität Graz eine Software zur Verwaltung gentechnischer Konstrukte realisiert.

Bei der Entwicklung der Software zeigte sich, dass eine immer wiederkehrende Adaptation, selbst im Aufbau der Software nötig war. Der Stellenwert von Details, welche am Anfang des Entwicklungsprozesses z.B. nicht als wichtig erkannt wurden, änderte sich im Laufe der Entwicklung des Programmes. So war der Aufbau der Software zu Beginn des Prozesses nicht deterministisch definiert. Ein interakiver und agiler Software-Entwicklungsprozess wurde angewendet.

Mit der Zeit kristallisierte sich eine Client-Software, die den Ansprüchen entspricht und gegebenenfalls Erweiterungsmöglichkeiten bietet.

Auf diesen Stand der Client-Software ist es nun möglich, dass der aktuelle Source-Code inklusive der Client-Software dem Institut für Biophysik übergeben werden kann. Die modulare und erweiterungsfähige Struktur der Programmierung erlaubt es jederzeit, dass die Client-Software verbessert und erweitert werden kann

## *4.1. Implementierung / Kompatibilität / Erweiterungsmöglichkeiten*

Da vom Institut für Biophysik keine spezielle Vorgabe in Bezug auf die Umsetzung der Software war, wurde eine Server-Client-Lösung umgesetzt. Es wäre auch möglich gewesen, den Client in einer ASP.NET Webseite unterzubringen. Dies hätte aber zur Folge gehabt, dass ein (zumeist gebührenpflichtiger) ASP.NET Hosting-Provider im Internet bzw. ein eigener Server am Institut für Biophysik vonnöten gewesen wäre.

Auch wäre es möglich gewesen, die Software in der Programmiersprache Java zu realisieren. Der Vorteil wäre unter anderem die plattform-unabhängige Verwendbarkeit der Software gewesen. Da aber am Institut nur Microsoft-Windows Rechner verwendet werden, war dies nie gefordert. Es wäre ein Nachteil gewesen, dass die Entwicklungsumgebung Microsoft Visual Studio 2013 Professional nicht nutzbar gewesen wäre. Diese zeichnet sich vor allem durch einfache Programmierung im Sinne von „Drag&Drop“ aus.

Um die Software einfacher zu erweitern und keine neue Kompilierung der Software zu benötigen, wäre es auch eine Möglichkeit gewesen, die Inhalte der Tabs in einzelne Programmbibliotheken (.DLL) auszulagern. Bei dieser Umsetzung müsste der Client nach diesen Bibliotheken suchen (die ein vorgegebenes Interface implementieren) und diese dann in die Oberfläche einbinden.

# 5. Literaturverzeichnis

1. File:DNA replication en.svg - Wikimedia Commons; 2016 [cited 2016 May 9]. Available from: URL: https://commons.wikimedia.org/wiki/File:DNA_replication_en.svg.

2. Gassen HG, Martin A, Sachse GE. Der Stoff aus dem die Gene sind: Bilder und Erklärungen zur Gentechnik. [3. Aufl.]. Frankfurt a.M. [etc.]: Campus Verl; 1990.

3. Passarge E. Taschenatlas der Genetik. 2., vollst. überarb. und erw. Aufl. Stuttgart [u.a.]: Thieme; 2004.

4. Passarge E, Wirth J. Taschenatlas Humangenetik. 3., vollst. überarb. Aufl. Stuttgart: Thieme; 2008.

5. Robinson TR, Schneider JH. Genetik für Dummies: [von Mendel bis Dolly - alles über die Vererbungslehre]. 1. Aufl. Weinheim: Wiley-VCH Verl.; 2006. (Für Dummies).

6. Polylinker; 2015 [cited 2015 Oct 22]. Available from: URL: https://de.wikipedia.org/w/index.php?oldid=130727475.

7. Vektor (Gentechnik); 2015 [cited 2015 Oct 22]. Available from: URL: https://de.wikipedia.org/w/index.php?oldid=139277135.

8. Servers - Patrycjusz Adamowicz 99195143; 2016 [cited 2016 Apr 29]. Available from: URL: https://sites.google.com/site/crcyclepma/home/servers.

9. 114 rv1, Postel, J, Reynolds. File Transfer Protocol; 2015 [cited 2015 Jul 6]. Available from: URL: https://tools.ietf.org/html/rfc959.

10. SQL Server Express-Edition | Microsoft [cited 2016 Mar 16]. Available from: URL: https://www.microsoft.com/de-de/server-cloud/products/sql-server-editions/sql-server-express.aspx.

11. FileZilla - The free FTP solution [cited 2015 Jul 6]. Available from: URL: https://filezilla-project.org/index.php.

12. Kurose JF, Ross KW. Computernetzwerke: Der Top-Down Ansatz. 4. Aufl. München: Pearson; 2008. (IT - Informatik).

13. Microsoft SQL Server; 2015 [cited 2015 Jul 6]. Available from: URL: https://de.wikipedia.org/w/index.php?oldid=142233118.

14. Oppel AJ, Sheldon R. SQL: A beginner's guide. - Description based on print version record. - Previous ed.: Robert Sheldon's name appeared first on t.p. - Includes index. 3rd ed. New York: McGraw-Hill; 2009.

15. CodeProject. CPOL: Code Project Open License - CodeProject [cited 2016 Mar 15]. Available from: URL: http://www.codeproject.com/info/cpol10.aspx.

16. metastruct. Simple C# FTP Class - CodeProject [cited 2016 Mar 15]. Available from: URL: http://www.codeproject.com/Tips/443588/Simple-Csharp-FTP-Class.

17. Globally Unique Identifier; 2016 [cited 2016 Mar 16]. Available from: URL: https://de.wikipedia.org/w/index.php?oldid=148002542.

18. SQL CREATE VIEW, CREATE OR REPLACE VIEW, DROP VIEW Statements; 2015 [cited 2015 Nov 21]. Available from: URL: http://www.w3schools.com/sql/sql_view.asp.

19. How To: Protect From SQL Injection in ASP.NET [cited 2016 Mar 15]. Available from: URL: https://msdn.microsoft.com/en-us/library/ff648339.aspx.

# 6. Anhang

## 6.1. Kommentierter C# Source-Code des Tabs ‚Batches‘

```
using BP_VGK.Common;
using BP_VGK.Data.MSSQL;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace BP_VGK.VerwaltungGentechKonstrukte.UserControls {

    public partial class BatchesUserControl : UserControl {

        //speichert die GUID des Datensatzes, der sich aktuell in den Eingabefeldern
befindet
        private Guid _idToUpdate = Guid.Empty;

        //Diese GUID wird benötigt, damit der aktuelle Datensatz markiert bleibt auch
wenn sich die Datenbank ändert.
        //ohne diesem würde sich bei einer Datenbank-Aktualisierung das DataGridView
ändern und es wäre danach ein falscher (zumeist der erste) Datensatz markiert
        private Guid overruleSelectedRow = Guid.Empty;

        public BatchesUserControl() {
            InitializeComponent();

            //Mithilfe dieser Methoden werden die Tooltips zu den einzelnen Controls
gesetzt
            InternalTools.SetToolTip(tbBatchNo, ToolTip_Constants.Batches.TB_Batch_No);
            InternalTools.SetToolTip(tbFreezerNo,
ToolTip_Constants.Batches.TB_Freezer_No);
            InternalTools.SetToolTip(cbAssociatedConstruct,
ToolTip_Constants.Batches.CB_Assoc_Construct);
            InternalTools.SetToolTip(tbNote, ToolTip_Constants.Batches.TB_Note);
            InternalTools.SetToolTip(cbTypeOfBatch,
ToolTip_Constants.Batches.CB_Type_of_Batch);
            InternalTools.SetToolTip(dtpCreated, ToolTip_Constants.Batches.DTP_Created);
            InternalTools.SetToolTip(tbOwner, ToolTip_Constants.Batches.TB_Owner);
            InternalTools.SetToolTip(btAddNewBatch,
ToolTip_Constants.Batches.BT_ADD_new_batch);
            InternalTools.SetToolTip(btUpdateExistingBatch,
ToolTip_Constants.Batches.BT_Update_Existing_batch);
            InternalTools.SetToolTip(btClearFields,
ToolTip_Constants.Batches.BT_Clear_Fields);
            InternalTools.SetToolTip(tbFilter, ToolTip_Constants.Batches.TB_Filter);
            InternalTools.SetToolTip(btClearFilter,
ToolTip_Constants.Batches.BT_Clear_Filter);

            //wählt als standard-eintrag den DNA-Eintrag beim typ des batches aus
            if (cbTypeOfBatch.Items.Count > 0)
                cbTypeOfBatch.SelectedIndex = 0;

            //setzt die aktuelle zeit beim erstelldatum eines neuen Batches
            dtpCreated.Value = DateTime.Now;
            //setzt das gewünschte Datumsformat für die Datums-Anzeige beim Erstelldatum
            string dateFormat =
INISettings.GetInstance(SettingsType.MainSettings).Read("Formatting",
"BatchesDateFormat");
            if (String.IsNullOrEmpty(dateFormat) == false) {
                dtpCreated.Format = DateTimePickerFormat.Custom;
                dtpCreated.CustomFormat = dateFormat;
            }
```

```
            //hängt sich an alle benötigten Events an, die Bescheid geben falls sich in
der Datenbank etwas ändert
            DataProvider.GetInstance().BatchesChanged +=
BatchesUserControl_BatchesChanged;
            //bei einer änderung der konstrukte wird auch die Methode zur aktualisierung
des datagridview aufgerufen... da sich ja der construct-name in diesem DGV befindet
            DataProvider.GetInstance().ConstructsChanged +=
BatchesUserControl_BatchesChanged;
            DataProvider.GetInstance().ConstructsChanged +=
BatchesUserControl_ConstructsChanged;

            //zur erstmaligen Auffüllung aller Listen, Datagridviews werden die Methoden
manuell aufgerufen
            BatchesUserControl_ConstructsChanged(null, null);
            BatchesUserControl_BatchesChanged(null, null);
        }

        /// <summary>
        /// somit lässt sich von extern der Filter setzen
        /// In diesem Fall nach Doppelklick auf einen Batch im Constructs-Tab
        /// Damit nur dieser angezeigt wird
        /// </summary>
        /// <param name="text"></param>
        internal void SetFilter(string text) {
            tbFilter.Text = text;
        }

        /// <summary>
        /// wenn sich etwas in der batches-tabelle ändert wird das datagridview
aktualisiert
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void BatchesUserControl_BatchesChanged(object sender, EventArgs e) {
            this.BeginInvokeIfRequired(new Action(() => {
                //wenn von einer anderen stelle im Programm eine selektierte Zeile
vorgegeben ist
                //z.b. die ID der neu angelegten Zeile (damit diese dann sofort nach
aktualisierung markiert wird)
                //wird diese ID in overruleSelectedRow zwischengespeichert
                Guid selGuid = overruleSelectedRow;
                //wenn keine externe ID vorgegeben ist wird die ID des gerade
ausgewählten Datensatzes abgespeichert
                if (selGuid == Guid.Empty)
                    selGuid = Tools.GetIDOfSelectedRow(ref dataGridView1, 0);
                //zwischenspeicher wird geleert
                overruleSelectedRow = Guid.Empty;
                //Das Datagridview wird manuell befüllt - in diesem Fall nur die
Datensätze die zu diesem Benutzer gehören

this.batches_Constructs_Owner_ViewTableAdapter.FillBy(this._BP_VGK_DatabaseDataSet.Batch
es_Constructs_Owner_View, UserManager.UserId);
                //Es wird eine Filterung der nun vorhandenen Datensätze durchgeführt,
mit Beachtung der aktuellen Filtereingabe (Text)
                //und falls die selektierte Guid in der angezeigten Menge auch angezeigt
wird (also der dazugehörige Datensatz)
                //wird dieser markiert
                filterDataSet(selGuid);
            }));
        }

        /// <summary>
        /// wenn sich die Konstrukte-Tabelle in der Datenbank aktualisiert wird die
combobox im eingabe-bereich aktualisiert
        /// </summary>
```

```
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void BatchesUserControl_ConstructsChanged(object sender, EventArgs e) {
            this.BeginInvokeIfRequired(new Action(() => {
                Guid selId = Guid.Empty;
                if (this.cbAssociatedConstruct.SelectedItem != null &&
cbAssociatedConstruct.SelectedItem is Construct)
                    selId = (this.cbAssociatedConstruct.SelectedItem as Construct).ID;
                updateEntriesinCBConstructs(selId);
            }));
        }

        /// <summary>
        /// sammmelt alle Daten die für einen neuen Batch erforderlich sind.
        /// Stellt ein Batch-Objekt zusammen und fügt dies in die Datenbank ein
        ///
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btAddNewBatch_Click(object sender, EventArgs e) {
            Batch newBatch = new Batch();
            newBatch.ID = Guid.Empty;
            newBatch.BatchID = tbBatchNo.Text;
            newBatch.FreezerNumber = tbFreezerNo.Text;
            newBatch.Information = tbNote.Text;
            if (cbAssociatedConstruct.SelectedItem != null &&
cbAssociatedConstruct.SelectedItem is Construct)
                newBatch.ConstructId = (cbAssociatedConstruct.SelectedItem as
Construct).ID;
            newBatch.Created = dtpCreated.Value;
            newBatch.OwnerId = UserManager.UserId;
            newBatch.TypeOfBatch = cbTypeOfBatch.SelectedItem.ToString();
            //speichert die neue ID zwischen damit der bei der aktualisierung des
Datagridview dieser sofort markiert wird
            Guid newId = DataProvider.GetInstance().AddOrUpdateBatch(newBatch);
            overruleSelectedRow = newId;
            //löscht daraufhin alle felder - neue eingabe somit möglich
            clearFields();
        }

        /// <summary>
        /// bereitet die oberfläche auf eine neue datensatz-eingabe vor
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btClearFields_Click(object sender, EventArgs e) {
            clearFields();
        }

        /// <summary>
        /// löscht den filter-text
        ///
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btClearFilter_Click(object sender, EventArgs e) {
            tbFilter.Text = string.Empty;
        }

        /// <summary>
        /// Aktualisiert den Datensatz
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btUpdateExistingBatch_Click(object sender, EventArgs e) {
            if (_idToUpdate != Guid.Empty) {
```

```
                Batch a = new Batch();
                a.ID = _idToUpdate;
                a.BatchID = tbBatchNo.Text;
                a.FreezerNumber = tbFreezerNo.Text;
                a.Information = tbNote.Text;
                a.Created = dtpCreated.Value;
                if (cbAssociatedConstruct.SelectedItem != null &&
cbAssociatedConstruct.SelectedItem is Construct)
                    a.ConstructId = (cbAssociatedConstruct.SelectedItem as
Construct).ID;
                a.TypeOfBatch = cbTypeOfBatch.SelectedItem.ToString();
                if (tbOwner.Tag != null && tbOwner.Tag is Guid) {
                    a.OwnerId = (Guid)tbOwner.Tag;
                    a.OwnerName = UserManager.Username;
                }
                Guid id = DataProvider.GetInstance().AddOrUpdateBatch(a);
                overruleSelectedRow = id;
                if (id == a.ID)
                    clearFields();
            }
        }

        /// <summary>
        /// löscht alle felder und bereitet die Oberfläche auf die Eingabe eines neuen
Datensatzes vor
        /// </summary>
        private void clearFields() {
            _idToUpdate = Guid.Empty;
            tbBatchNo.Text = string.Empty;
            tbFreezerNo.Text = string.Empty;
            tbNote.Text = string.Empty;
            tbOwner.Text = string.Empty;
            tbOwner.Tag = null;
            cbTypeOfBatch.SelectedIndex = 0;
            dtpCreated.Value = DateTime.Now;
            updateEntriesinCBConstructs(Guid.Empty);
        }

        /// <summary>
        /// Ruft, wenn in der richtigen spalte geklickt wurde, die Datei Doku auf
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
            if (e.RowIndex >= 0 && e.ColumnIndex == 8) {
                Guid id = Tools.GetIDOfSelectedRow(ref dataGridView1, 0);
                FileDocumentationBrowserForm fdbf = new FileDocumentationBrowserForm(id,
"Batch \"" + dataGridView1.Rows[e.RowIndex].Cells[1].Value.ToString() + "\"");
                if (fdbf.Reachable)
                    fdbf.ShowDialog(this);
            }
        }

        /// <summary>
        /// schreibt auf den File doku button 'Browse...' rauf und formatiert das datum
in der created-spalte richtig
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dataGridView1_CellFormatting(object sender,
DataGridViewCellFormattingEventArgs e) {
            if (e.RowIndex >= 0 && e.ColumnIndex == 8) {
                e.Value = "Browse...";
            }
            if (e.RowIndex >= 0 && e.ColumnIndex == 3) {
```

```
                var x = e.Value;
                if (x != null && !String.IsNullOrEmpty(x.ToString())) {
                    string s = x.ToString();
                    DateTime erg = DateTime.Now;
                    if (DateTime.TryParse(s, out erg)) {
                        string format =
INISettings.GetInstance(SettingsType.MainSettings).Read("Formatting",
"BatchesDateFormat");
                        if (String.IsNullOrEmpty(format)) {

INISettings.GetInstance(SettingsType.MainSettings).Write("Formatting",
"BatchesDateFormat", "d");
                            format = "d";
                        }
                        try {
                            e.Value = erg.ToString(format);
                        } catch (Exception) { }
                    }
                }
            }
        }

        /// <summary>
        /// wenn auf den zeilen-kopf rechts geklickt wird soll ein ToolStripMenuItem
erscheinen wo man danach die aktuelle Zeile (datensatz) löschen kann
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dataGridView1_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e) {
            if (e.RowIndex >= 0) {
                if (e.Button == System.Windows.Forms.MouseButtons.Left) {
                    dataGridView1_SelectionChanged(sender, e);
                }
                if (e.Button == System.Windows.Forms.MouseButtons.Right) {
                    ContextMenuStrip cms = new ContextMenuStrip();
                    ToolStripMenuItem tsi = new ToolStripMenuItem();
                    tsi.Text = "Delete Batch " +
dataGridView1.Rows[e.RowIndex].Cells[1].Value.ToString() + " ?";
                    tsi.Tag = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
                    cms.Items.Add(tsi);
                    cms.ItemClicked += (sender1, evArgs) => {
                        object tag = evArgs.ClickedItem.Tag;
                        if (tag != null && tag is string) {
                            Guid batchToDelete = new Guid(tag.ToString());
                            DataProvider.GetInstance().DeleteBatch(batchToDelete);
                            clearFields();
                        }
                    };
                    cms.Show(Cursor.Position);
                }
            }
        }

        /// <summary>
        /// wenn sich die Selection im datagridview ändert soll dieser datensatz im
eingabebereich angezeigt werden
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dataGridView1_SelectionChanged(object sender, EventArgs e) {
            if (dataGridView1.SelectedRows != null && dataGridView1.SelectedRows.Count >
0) {
                Guid id =
Tools.MyTryParse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());
                Batch selBatch = DataProvider.GetInstance().GetBatch(id);
```

```
                if (selBatch != null) {
                    _idToUpdate = selBatch.ID;
                    tbBatchNo.Text = selBatch.BatchID;
                    tbFreezerNo.Text = selBatch.FreezerNumber;
                    tbNote.Text = selBatch.Information;
                    updateEntriesinCBConstructs(selBatch.ConstructId);
                    dtpCreated.Value = selBatch.Created;
                    User u = DataProvider.GetInstance().GetUser(selBatch.OwnerId);
                    tbOwner.Text = u != null ? u.Username : string.Empty;
                    if (String.IsNullOrEmpty(tbOwner.Text) == false)
                        tbOwner.Tag = selBatch.OwnerId;
                    else
                        tbOwner.Tag = null;
                    cbTypeOfBatch.SelectedItem = selBatch.TypeOfBatch;
                }
            }
        }

        /// <summary>
        /// filtert das dgv nach dem Text der in tbFilter steht.. und falls die aktuelle
selected row sich in dieser auswahl befindet soll diese markiert bleiben
        /// </summary>
        /// <param name="idOfSelectedRow"></param>
        private void filterDataSet(Guid idOfSelectedRow) {
            this.BeginInvokeIfRequired(new Action(() => {
                Tools.SearchInDatagridView(tbFilter.Text, ref dataGridView1, 0, new
int[] { 1, 2, 3, 4, 5, 6, 9 }, idOfSelectedRow);
            }));
        }

        /// <summary>
        /// aktualisiert die Buttons AddNew und UpdateExisting je nach Konstellation
        /// also ob sich in den eingabefeldern ein text befindet und ob eine aktuelle ID
eines datensatzes global gespeichert ist
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void inputData_TextChanged(object sender, EventArgs e) {
            Tools.HandleButtons(new string[] { tbBatchNo.Text, this.tbFreezerNo.Text,
tbNote.Text }, this._idToUpdate, ref btAddNewBatch, ref btUpdateExistingBatch);
        }

        /// <summary>
        /// bei änderung des filter-texts sollen nur die passenden datensätze im DGV
angezeigt werden
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void tbFilter_TextChanged(object sender, EventArgs e) {
            filterDataSet(Tools.GetIDOfSelectedRow(ref dataGridView1, 0));
        }
        /// <summary>
        /// Wenn der batch einem anderen user gehört und man selbst admin ist wird
mithilfe dieser methode der besitzer auf sich selbst geändert
        /// update-batch hier nicht vergessen
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void tbOwner_MouseDoubleClick(object sender, MouseEventArgs e) {
            if (DataProvider.GetInstance().IsAdminUser(UserManager.UserId)) {
                if (tbOwner.Tag is Guid || tbOwner.Tag == null) {
                    Guid idOfOwner = tbOwner.Tag != null ? (Guid)tbOwner.Tag :
Guid.Empty;
                    if (UserManager.UserId != idOfOwner) {
```

```
                        var mb = MessageBox.Show(this, "Do you really want to take
ownership of this batch?", "Taking Ownership of batch", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
                        if (mb == DialogResult.Yes) {
                            tbOwner.Text = UserManager.Username;
                            tbOwner.Tag = UserManager.UserId;
                        }
                    }
                }
            }
        }

        /// <summary>
        /// aktualisiert die combobox wo die liste aller konstrukte angezeigt werden
        /// </summary>
        /// <param name="selectedConstructId">falls sich ein construct mit dieser id
findet wird dieses als selecteditem gesetzt</param>
        private void updateEntriesinCBConstructs(Guid selectedConstructId) {
            cbAssociatedConstruct.Items.Clear();
            List<Construct> li = new List<Construct>();
            li.AddRange(DataProvider.GetInstance().GetAllConstructsSimple());
            cbAssociatedConstruct.Items.AddRange(li.ToArray());
            cbAssociatedConstruct.SelectedItem = li.Where(yx => yx.ID ==
selectedConstructId).FirstOrDefault();
        }
    }
```