Hochschule Ostfalia Wolfsburg

Team Tierfutter

Projekt Mini-Zoo

29.5.2020

Inhalt

Abbildungsverzeichnis

Einleitung

In dieser Projektarbeit soll eine Datenbank entworfen werden. Die Datenbank soll aus mindestens zwei Tabellen bestehen und es sollen zwei verschiedene Anwendungsfälle mit unterschiedlichen Sichten entworfen werden.

Für dieses Projekt wird als Umgebung ein Zoo gewählt. Die Datenbank soll die Arbeit der Tierpfleger sowie der Zooverwaltung erleichtern indem Informationen zu den Tieren und den Lagerbeständen für beide Anwender übersichtlich dargestellt werden.

Rollenverteilung

Das Team besteht aus 4 Teilnehmern. Zwei Personen bilden das Architektur Team. Ihre Aufgabe ist es die Anforderungen für das Projekt zu spezifizieren und daraus ein ER-Diagramm zu erzeugen. Aus dem ER-Diagramm wird im Anschluss ein relationales Datenbankmodell entwickelt welches im Anschluss umgesetzt werden kann.

Die Entwicklung wird von einer Person übernommen. Ziel der Entwicklung ist es das in der Architektur entstandene relationale Datenmodell in einer Software umzusetzen. Außerdem setzt die Entwicklung die weiteren Anforderungen wie beispielsweise Abfragen an die Datenbank in der Software um.

Die vierte Person bildet das Testing Team. Es überprüft ob die Anforderungen der Architektur Gruppe von der Entwicklungsgruppe korrekt umgesetzt wurden. Die Testfälle werden entsprechend dokumentiert.

Entwicklungszyklus

Das Team entwickelt in einem agilen SCRUM. In einem zweitägigen Rhythmus werden Arbeitsziele definiert, die bis zu dem nächsten Meeting umgesetzt werden sollen. Dadurch kann agil auf Probleme reagiert werden. Außerdem ermöglicht das Modell allen Gruppen bereits zu beginn des Projektes mit der Arbeit zu beginnen. Durch die regelmäßigen Meetings können Sprint-Ziele jederzeit flexibel angepasst werden.

Softwarearchitektur

Abgrenzung des Projektrahmens

Im Rahmen des Projekts wird der Fokus auf die Funktionalität der Datenbank bezüglich des Inhalts und der relevanten Queries gelegt. Die Verwendung der Software soll über angelegte Benutzer auf einer Benutzeroberfläche erfolgen, deren Implementierung allerdings nicht vorgenommen wird. Aus diesem Grund werden die Anforderungen an die Software im Folgenden aus funktionaler Sicht gestellt und kein Bezug auf das Aussehen der Benutzeroberfläche genommen. Im Kapitel „Ausblick“ wird dann darauf eingegangen, inwiefern man die Benutzeroberfläche noch mit einbinden könnte.

Anforderungen an die Software

Im ersten Schritt müssen die für die Software relevanten Informationen aus der Kurzbeschreibung des Projekts erfasst werden. Damit werden dann konkrete Anforderungen an die Software hinsichtlich der benötigten Funktionalität formuliert. Die Software ist dabei als „Black-Box“ zu sehen, die eine Eingabe oder Anfrage erhält und dazu eine entsprechende Ausgabe liefert. Die Anforderungen werden zur besseren Übersicht und Zurückverfolgbarkeit mit einer eindeutigen Nummerierung versehen.

Zuerst ist eine Unterteilung in die verschiedenen Anwendungsfälle erforderlich. Außerdem muss festgelegt werden, welche Informationen die Datenbank am Ende enthalten soll. Danach werden die einzelnen Anwendungen aus Sicht des jeweiligen Benutzers behandelt.

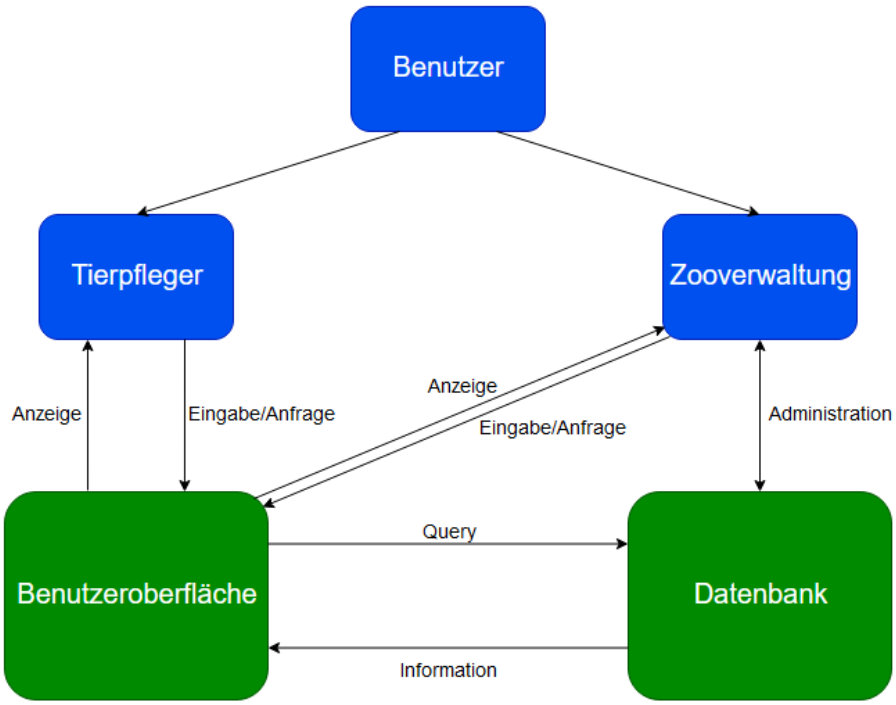

Abbildung 1: Schema Use-Case

1: Unterteilung in Anwendungsfälle (Use-Cases):

Die Verwendung der Datenbank soll aus zwei verschiedenen Sichten auf die Daten erfolgen. Der erste Anwendungsfall ist aus Sicht des Tierpflegers, der zweite aus Sicht der Verwaltung des Zoos.

2: Enthaltene Informationen:

2.1: Die Datenbank muss für jedes individuelle Tier die zugehörigen Eigenschaften Tierart, Name, Geburtsdatum, Geschlecht und eine eindeutige Identifikation (Tier-ID) enthalten.

2.2: Die Datenbank muss die eingelagerten Futtersorten mit der aktuellen Anzahl der Lagereinheiten (Ist-Wert), einem oberen Grenzwert (Maximalwert) und einem unteren Grenzwert (Minimalwert) enthalten. Zusätzlich soll eine Beschreibung der Lagereinheit (z. B. 5kg Säcke) verfügbar sein.

2.3: Die Datenbank muss die Fütterungen der Tiere mit einer Angabe des Wochentags, jeweiliger Futtermenge, Uhrzeit und einer eindeutigen Identifikation (Fütterung-ID) enthalten. Dabei soll jedem Tier eine Fütterung und jeder Fütterung eine Futtersorte zugeordnet sein.

3: Anwendungen:

3.1: Der Tierpfleger soll Fütterungspläne für die einzelnen Wochentage erstellen können. Dazu muss eine nach der Uhrzeit sortierte Liste mit Einträgen zu allen Fütterungen ausgegeben werden, die am jeweiligen Wochentag stattfinden. Die Liste soll dabei die Tierart, den Namen und die ID der Tiere sowie die zugehörige Futtermenge und Uhrzeit enthalten.

3.2: Der Tierpfleger soll eine bestimmte Tierart als Suchbegriff eingeben können. Bei gültiger Eingabe sind die Einträge aller dieser Art angehörenden Individuen mit den Informationen ID, Name, Alter, Geschlecht und Futtermenge in Form einer Liste anzuzeigen. Das Alter soll dabei aus dem Geburtsdatum abgeleitet werden.

3.3: Der Tierpfleger darf keinen Zugriff auf die Bearbeitung der Einträge in der Datenbank haben.

3.4: Die Zooverwaltung soll administrative Rechte besitzen. Es sollen einzelne Einträge bearbeitet, hinzugefügt und entfernt werden können.

3.5: Es soll auf Anfrage der Zooverwaltung eine Einkaufsliste generiert werden. Dazu muss bei jeder Futtersorte die Differenz zwischen hinterlegtem Maximalwert und aktueller Anzahl der Lagereinheiten gebildet und dieser Wert dann zusammen mit der zugehörigen Futtersorte als Liste ausgegeben werden.

3.6: Wenn die aktuelle Anzahl der Lagereinheiten einer Futtersorte den angegebenen Minimalwert unterschreitet, soll eine Warnmeldung mit der betroffenen Futtersorte für die Zooverwaltung erzeugt werden.

Entity-Relationship-Modell (ERM)

Das Entity-Relationship-Modell dient dazu, einen relevanten Ausschnitt aus der realen Welt, der dann als sogenannte „Mini-Welt“ in der Datenbank abgebildet wird, zu bestimmen und darzustellen. Dazu wird eine Grafik (ER-Diagramm) sowie eine Beschreibung der darin enthaltenen Elemente verwendet. Nun muss also aus den gegebenen Sachverhalten und Anforderungen ein ER-Diagramm erstellt werden. Dazu wird die Chen-Notation benutzt. Im ersten Schritt sind die sogenannten Entitätstypen zu identifizieren. Ein Entitätstyp umfasst eine Menge von gleichartigen Objekten der realen Welt (Entitäten), die allerdings nur über ihre Eigenschaften (Attribute) beschrieben werden können. Die Beziehungen zwischen den Entitätstypen werden Relationen genannt. Sie zeigen die Zusammenhänge auf.

Im konkreten Fall des Zoos wird der Tierpfleger betrachtet, der die einzelnen Tiere füttern muss und die Zooverwaltung, die für das Futterlager zuständig ist. Da ein Fütterungsplan erstellt werden soll, muss es Informationen zu den einzelnen Fütterungen der Tiere geben. Das Futter für die Fütterungen wird vom Tierpfleger aus dem Futterlager entnommen. Außerdem sollen Informationen zu den einzelnen Tieren und die Bestände der eingelagerten Futtersorten enthalten sein. Daraus ergeben sich die Entitätstypen Tier, Fütterung und Futterlager, die über ihre relevanten Eigenschaften (Attribute) beschrieben werden können.

Die Entitäten des Entitätstyps Tier weisen die Attribute Art, Name, Geschlecht und Geburtsdatum auf. Zusätzlich wird eine eindeutige Identifikation, ein sogenanntes Schlüsselattribut, in Form einer Tier-ID vergeben. In diesem Fall ist es nur ein einfacher Schlüssel, da er nur aus einem Attribut besteht. Man könnte hier auch einen zusammengesetzten Schlüssel aus den Attributen Art und Name verwenden, wenn man sich darauf einigt, dass ein bestimmter Name nur einmal pro Tierart vergeben werden darf. In unserem Fall haben wir uns allerdings auf eine Identifikation der Tiere durch eine eindeutige Nummer beschränkt.

Die individuellen Fütterungen der einzelnen Tiere sollen ebenfalls durch eine eindeutige Fütterungs-ID unterschieden werden. Die weiteren Attribute einer Fütterung sind die Menge, die verfüttert wird, und die zeitliche Angabe über Uhrzeit und Wochentag, also wann diese Fütterung stattfinden soll.

Das Futterlager enthält die verschiedenen Sorten des eingelagerten Futters. Da die jeweilige Sorte schon eine eindeutige Beschreibung der Entität darstellt, wird dieses Attribut als das Schlüsselattribut verwendet. Auch dieses entspricht einem einfachen Schlüssel. Jede Futtersorte erhält außerdem die Attribute Lagereinheit, Maximum und Minimum. Die Lagereinheit ist dabei ein zusammengesetztes Attribut aus der Anzahl der Einheiten, der Menge pro Einheit und einer Beschreibung der Einheit. Diese Einteilung soll an einem kurzen Beispiel verdeutlicht werden. Die Futtersorte Bananen wird in der Einheit 5kg Boxen eingelagert. Davon befinden sich zurzeit 10 Boxen im Lager. Die Anzahl der Einheiten wäre damit 10 und die Menge pro Einheit 5. Als Beschreibung dient dann „kg pro Box“. Die Trennung der Menge pro Einheit und Beschreibung ist für spätere Rechenoperationen nötig. Die Angabe eines maximalen und minimalen Werts soll der Zooverwaltung als Orientierung dienen. Das Maximum gibt an, wie viele Einheiten der jeweiligen Futtersorte maximal im Lager enthalten sein sollen. Das Minimum hingegen steht für einen unteren Grenzwert, bei dessen Unterschreitung Futter der jeweiligen Sorte zwingend nachbestellt werden muss.

Die Beziehungen zwischen den Entitätstypen werden wie folgt definiert: Ein Tier „bekommt“ eine Fütterung und die Fütterung „benötigt Futter aus“ dem Futterlager. Die Tiere sollen nicht zwingend nur einmal am Tag oder jeden Tag gefüttert werden, also muss wie schon beschrieben eine Einteilung mit Wochentag und Uhrzeit vorgenommen werden. Einem Tier werden dann mehrere Fütterungen in der Woche zugeordnet. Von daher ergibt sich eine Kardinalität von 1:N für die Beziehung zwischen Tier und Fütterung. Beide Entitätstypen nehmen total an der Relation teil, es muss also jedem Tier mindestens eine Fütterung und jeder Fütterung ein Tier zugeordnet sein. Die verschiedenen Fütterungen benötigen teilweise die gleiche Sorte Futter aus dem Lager, weshalb hier eine N:1 Beziehung zwischen Fütterung und Futterlager vorliegt. Hier nimmt der Entitätstyp Fütterung total an der Relation teil, da jede Fütterung irgendeine Sorte von Futter benötigt. Im Gegensatz dazu soll es eingelagerte Futtersorten geben, die noch keiner Fütterung zugeordnet sind.

Das fertige ER-Diagramm mit den beschriebenen Inhalten ist in der folgenden Abbildung zu sehen.

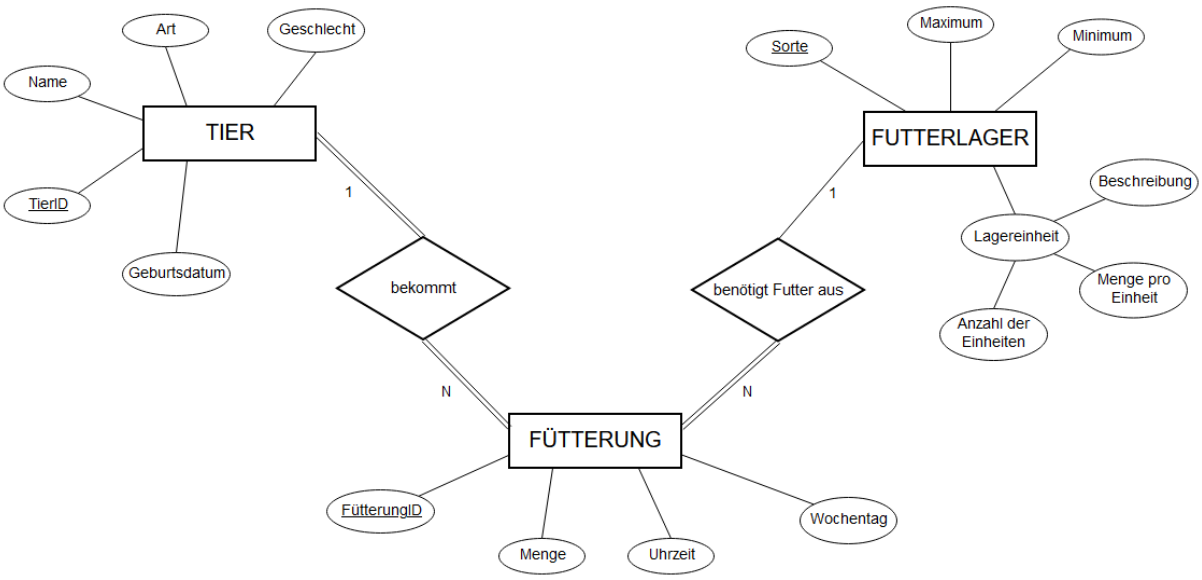

Abbildung 2: ER-Diagram

Überführung des ER-Schemas in ein relationales Schema

Der konzeptionelle Entwurf in Form des ER-Schemas beziehungsweise ER-Diagramms muss im nächsten Schritt möglichst verlustfrei in einen logischen Entwurf übersetzt werden. Aus dem sogenannten relationalen Schema lassen sich dann die benötigten Tabellen für die Datenbank gewinnen.

Die Übersetzung erfolgt in sieben Schritten. Zuerst werden die starken Entitäten aus dem ER-Diagramm behandelt. Für jede starke Entität wird ein gleichnamiges Relationenschema erzeugt, das alle atomaren Attribute der Entität beinhaltet. Das zusammengesetzte Attribut Lagereinheit wird dabei in seine atomaren Attribute zerlegt. Das jeweilige Schlüsselattribut aus dem ER-Diagramm wird als Primärschlüssel für das Relationenschema übernommen. Da alle vorhandenen Entitäten starke Entitäten sind, entfällt die Behandlung schwacher Entitäten an diesem Punkt und es wird mit der Behandlung der verschiedenen Beziehungen zwischen den Entitäten fortgefahren. In diesem Fall gibt es nur zwei Beziehungen der Form N:1 beziehungsweise 1:N. Die Relationenschemata der jeweils an der Relation beteiligten Entitäten müssen identifiziert und das N-seitige Relationenschema um einen Fremdschlüssel erweitert werden, der den Primärschlüssel des 1-seitigen Relationenschemas referenziert. Das Relationenschema zu Fütterung muss also um zwei Fremdschlüssel „TierID“ und „Sorte“ erweitert werden. Dadurch entsteht der gewünschte Zusammenhang zwischen dem individuellen Tier und seiner zugehörigen Futtersorte. Es bleiben nun noch die Behandlung nicht atomarer (mehrwertiger) Attribute und n-ärer Beziehungen. Diese kommen hier allerdings nicht vor. Im Folgenden sind die nun erzeugten Relationenschemata gezeigt. Jedes Schema entspricht dabei einer Tabelle für die Datenbank.

Tier

TierID	Art	Name	Geschlecht	Geburtsdatum

Fuetterung

FuetterungID	Wochentag	Uhrzeit	Menge	TierID	Sorte

Futterlager

Sorte	Maximum	Minimum	Anzahl Einheiten	Menge/Einheit	Beschreibung

Abbildung 3: relationales Schema

Normalisierung

Im letzten Schritt sollen die erzeugten Relationenschemata mithilfe von Normalformen untersucht werden. Durch die Normalformen wird ein objektiver Vergleich von Relationenschemata ermöglicht, der Aussagen hinsichtlich ihrer Qualität zulässt. Dabei sind vor allem die funktionalen Abhängigkeiten zwischen Mengen von Attributen innerhalb eines Relationenschemas interessant. Zusätzlich wird die Reduktion von Redundanzen und NULL-Werten in Tupeln sowie die Vermeidung der Erzeugung von unechten Tupeln bei der Verwendung von JOINs angestrebt.

Bei der Modellierung wurde schon darauf geachtet, diese Qualitätsmaße möglichst einzuhalten. Zum Beispiel wurden keine Attribute mit gleichem Namen verwendet außer für die Referenzierungen zwischen den Schemata. Dies verhindert das Entstehen unechter Tupel bei der Ausführung von JOINs mit Gleichheitsbedingungen über Attributen. Des Weiteren wurden keine Attribute aus getrennt zu modellierenden Entitäten in einem Relationenschema kombiniert, wodurch sich die Bedeutungen der einzelnen Schemata deutlich abgrenzen lassen. Dadurch treten im Idealfall mögliche transitive

Abhängigkeiten von Attributen nicht auf und es entstehen keine Probleme bei der Durchführung von Einfüge-, Lösch- und Update-Operationen.

Im Folgenden sollen die Schemata den Tests der ersten drei Normalformen unterzogen und im Anschluss noch auf die Boyce-Codd-Normalform eingegangen werden.

Die erste Normalform (1NF) besagt, dass die Attribute eines Relationenschemas nur atomare, also unteilbare, Werte enthalten dürfen. Diese Bedingung wurde schon bei der Übersetzung aus dem ER-Schema in das relationale Schema beachtet, wodurch alle Schemata ausschließlich atomare Attribute beinhalten. Sie sind dementsprechend alle in 1NF.

Die zweite Normalform (2NF) wird erreicht, wenn das Relationenschema in 1NF ist und kein nicht-primes Attribut von nur einem Teil des Primärschlüssels abhängt. Diese Bedingung setzt voraus, dass es einen aus mehreren Attributen zusammengesetzten Primärschlüssel gibt. Da alle betrachteten Relationenschemata nur einen „einfachen“ Primärschlüssel besitzen, sind sie automatisch alle in 2NF.

Zur Erfüllung der dritten Normalform (3NF) muss ein Relationenschema in 2NF sein und es darf kein nicht-primes Attribut transitiv vom Primärschlüssel abhängen. Es muss also geprüft werden, ob ein Nichtschlüsselattribut existiert, das von einem oder mehreren anderen Nichtschlüsselattributen funktional abhängig ist. Da schon beim Entwurf darauf geachtet wurde, dass die Relationenschemata jeweils genau einer Entität entsprechen, konnten hier direkt transitive Abhängigkeiten von Attributen vom Primärschlüssel vermieden werden. Die Schemata befinden sich folglich ohne Zerlegung in 3NF.

3NF-Relationenschemata können nun allerdings immernoch problematische Eigenschaften aufweisen. Die Boyce-Codd-Normalform stellt eine weitere Verschärfung der Anforderungen an die Schemata dar. Alle funktionalen Abhängigkeiten innerhalb eines Relationenschemas dürfen nur die Abhängigkeiten der Attribute vom Superschlüssel sein. Im Falle der betrachteten Relationenschemata entspricht der Superschlüssel jeweils dem Primärschlüssel, da dessen Wert für alle enthaltenen Tupel jeweils einzigartig ist. Es existieren in allen drei Schemata nur funktionale Abhängigkeiten der Attribute vom Superschlüssel, wodurch voneinander unabhängige Daten auch in separaten Relationenschemata gespeichert werden. Damit ist die Boyce-Codd-Normalform für alle drei betrachteten Schemata erfüllt.

Modulanforderungen

Die Anforderungen sind in zwei Module aufgeteilt.

Module Verwaltung

Info : - die Bearbeitungsfunktion wird von der Verwaltung genutzt ,um eine bestimmte Liste zu verwalten und Einträge zu bearbeiten.

MA-DB-3 : die Bearbeitungsfunktion nimmt als Parameter eine Liste `Verwaltung_Liste [ ] ` die Einträge hinzufügt und entfernt.

MA-DB-4 :Die Funktion Bestellliste soll eine Liste `Bestellungsliste [ ]` erstellen, die jeden Futtertyp `Futter_ID :int(11)` die Differenz zwischen dem gespeicherten Maximalwert `Max Einheit :int(11)` und den Anzahl der Lagereinheiten `Anzahl Einheiten : int(11)` mit dem jeweiligen Futtersorte ausgegeben werden .

Info : -Die Sortierfunktion soll eine Warnmeldung mit der betroffenen Futtersorte Wenn ein Lagerbestand einer Futtersorte den angegebenen Minimalwert unterschreitet.

MA-DB-5 : Die Meldungsfunktion soll eine Warnmeldung für die Zooverwaltung generiert werden, wenn die `Anzahl Einheiten :int (11)` in der Tabelle `Lager` für einen bestimmten Futtertyp `Futter_ID :int(11)` in der Tabelle `Futter` unter dem angegebenen Mindestwert `Min Einheit :int(11)` liegt .

Module Tierpfleger

Info: Die Funktion Tierfutter soll eine sortierte Liste Futter_zeit [ ] mit Einträge von allen Fütterungen zeigen .

MA-DB-1: In die Liste `Futter_zeit [ ] ` werden die Fütterungen mit einer Angabe des wochentags nach der *Fuetterungszeit* `Fuetterungszeit :time` mit dem Attributen `Tierart:varchar(50)` ,`Tier_ID: int(11)` ,`Name `und zugehörige `Futtermenge:int(11)` angezeigen .

MA-DB-2: mit dem Primärschlüssel `Tier_ID: int(11)` die suchfunktion sollte Informationen über ein bestimmtes Tier suchen und die andere Attribute vorstellen :`Futter_ID :int(11)` mit Futtermenge:int(11)` und `Fuetterungzeit :time` .

Entwicklung

Einleitung

In der Entwicklung unserer Datenbank werden die Anforderungen, die während des Designs der Architektur entstanden sind, umgesetzt. Während der Entwicklung der Datenbank kann durch einen entsprechenden SCRUM Rhythmus agil auf Fehler reagiert werden, die während der Architektur entstanden sind.

XAMPP

XAMPP ist ein Softwarepaket, welches von dem Entwickler Apache Friends entwickelt und vertrieben wird. In dem Paket ist ein Apache Server enthalten auf welchem eine MariaDB Datenbank läuft. Diese kann mittels phpMyAdmin erstellt und administriert werden.

Des Weiteren steht die FTP Datenübertragung Filezilla, sowie der Mailserver Mercury und der Java Server Apache Tomcat zur Verfügung. Diese Programme finden in unserem Anwendungsfall jedoch keine Verwendung.

Um die Datenbank nun zu erstellen und zu nutzen wird der entsprechende Apache Server und MariaDB gestartet. Über einen Internetbrowser lässt der Server lokale Server erreichen.

Tabellen erstellen

Um die Tabellen der Datenbank zu erstellen wird im Browser die Nutzeroberfläche phpMyAdmin genutzt. Das ER-Diagramm setzt 3 Tabellen vor. Diese werden mit den gezeigten Attributen erstellt.

Tier

Das Tier erhält nach dem ER- Diagramm die Attribute: *Tier_ID, Name, Geburtstdatum, Tierart* und *Geschlecht*. Die entsprechenden Dateiformate sind in Abbildung 4 zu sehen. Für das Attribut Tier_ID wird die Option AUTO_INCREMENT genutzt. Die ermöglicht ein automatisches erhöhen der ID beim Hinzufügen neuer Datensätze. Des Weiteren wurden die Attribute Name, Tierart und Geschlecht auf jeweils 50 und 20 Zeichen begrenzt. Der Primärschlüssel der Tabelle ist die ID.

	#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
☐	1	Tier_ID 🔑	int(11)			Nein	*kein(e)*		AUTO_INCREMENT	Bearbeiten Löschen Mehr
☐	2	Name	varchar(50)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten Löschen Mehr
☐	3	Geburtsdatum	date			Nein	*kein(e)*			Bearbeiten Löschen Mehr
☐	4	Tierart	varchar(50)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten Löschen Mehr
☐	5	Geschlecht	varchar(20)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten Löschen Mehr

Abbildung 4: Tabelle Tier, Struktur

Lager

Die Tabelle Lager erhält die Attribute: *Sorte, Einheit, Anzahl Einheiten, Menge/Einheit, Minimum* und *Maximum*. Da sich nicht zweimal das Selbe Futter im Lager befindet kann hier die Futtersorte als Primärattribut genutzt werden.

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	Sorte	varchar(50)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten Löschen Mehr
2	Beschreibung	varchar(20)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten Löschen Mehr
3	Anzahl Einheiten	int(11)			Nein	*kein(e)*			Bearbeiten Löschen Mehr
4	Menge/Einheit	int(11)			Nein	*kein(e)*			Bearbeiten Löschen Mehr
5	Min. Einheit	int(11)			Nein	*kein(e)*			Bearbeiten Löschen Mehr
6	Max Einheit	int(11)			Nein	*kein(e)*			Bearbeiten Löschen Mehr

Abbildung 5: Tabelle Lager, Struktur

Fuetterung

In der Tabelle der Fütterung werden die Attribute: *Fuetterungs_ID, Tier_ID, Sorte, Futtermenge (kg), Fuetterungszeit* und *Fuetterungstag* hinzugefügt. Die *Fuetterungs_ID* bildet den Primärschlüssel. Eine Besonderheit dieser Tabelle ist das Dateiformat der *Fuetterungszeit*. Es wird in dem Dateiformat *time* gearbeitet welches das Speichern einer Uhrzeit mit Stunden, Minuten und Sekunden innerhalb einer Zelle ermöglicht.

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	Fuetterungs_ID	int(11)			Nein	*kein(e)*		AUTO_INCREMENT	Bearbeiten Löschen Mehr
2	Tier_ID	int(11)			Nein	*kein(e)*			Bearbeiten Löschen Mehr
3	Sorte	varchar(50)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten Löschen Mehr
4	Futtermenge (kg)	int(11)			Nein	*kein(e)*			Bearbeiten Löschen Mehr
5	Fuetterungszeit	time			Nein	*kein(e)*			Bearbeiten Löschen Mehr
6	Fuetterungstag	varchar(20)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten Löschen Mehr

Abbildung 6: Tabelle Fuetterung, Struktur

Tabellen verknüpfen

Spalte	Interne Beziehung		
Fuetterungs_ID	tierfutter		
Tier_ID	tierfutter	tier	Tier_ID
Sorte	tierfutter	lager	Sorte
Futtermenge (kg)	tierfutter		
Fuetterungszeit	tierfutter		
Fuetterungstag	tierfutter		

Abbildung 7: Beziehungsansicht

Um die Tabellen mit den entsprechenden Beziehungen zu verknüpfen werden die Primärschlüssel in der Beziehungsansicht eingetragen. Wie in dem ER-Diagramm wird der Primärschlüssel *Tier_ID* der Tabelle *tier* in die Tabelle *fuetterung* eingebunden. Der Primärschlüssel *Sorte* der Tabelle *lager* wird ebenfalls in die Tabelle der *fuetterung* eingebunden.

Exportieren der Datenbank

Um die Datenbank nun dem Tester zur Verfügung zu stellen wird diese Exportiert. Dabei wird eine SQL Datei erzeugt in der alle Befehle eingetragen sind die das erzeugen der Datenbank in verschiedenen Programmen ermöglicht.

SQL-Abfragen erstellen

In den Anforderungen des Projektes wird ein Fütterungsplan gefordert. Dazu wird eine entsprechende SQL Abfrage erstellt.

```
    SELECT `fuetterung`.`Fuetterungstag`, `fuetterung`.`Fuetterungszeit`, `tier`.`Tierart`, `tier`.`Name`, `fuetterung`.`Futtermenge (kg)`, `lager`.`Sorte`
FROM `fuetterung`
    , `tier`
    , `lager`
WHERE `fuetterung`.`Sorte` = `lager`.`Sorte`
AND `fuetterung`.`Tier_ID` = `tier`.`Tier_ID`
ORDER BY
     CASE
          WHEN `fuetterung`.`Fuetterungstag` = 'Montag' THEN 1
          WHEN `fuetterung`.`Fuetterungstag` = 'Dienstag' THEN 2
          WHEN `fuetterung`.`Fuetterungstag` = 'Mittwoch' THEN 3
          WHEN `fuetterung`.`Fuetterungstag` = 'Donnerstag' THEN 4
          WHEN `fuetterung`.`Fuetterungstag` = 'Freitag' THEN 5
          WHEN `fuetterung`.`Fuetterungstag` = 'Samstag' THEN 6
          WHEN `fuetterung`.`Fuetterungstag` = 'Sonntag' THEN 7
     END ASC,
     `fuetterung`.`Fuetterungszeit`ASC
```

Abbildung 8: SQL-Futterplan

Der Befehl SELECT selektiert die Benötigten Einträge der Datenbank. Mit der Bedingung WHERE werden nur die miteinander verknüpften Datensätze angezeigt. Um den Fütterungsplan nun korrekt sortiert anzeigen zu können wird der Befehl ORDER BY genutzt. Mittels CASE werden die Wochentage als Zahlen genutzt und nach Größe Aufsteigend sortiert. Die Einträge werden ein zweites mal nach Uhrzeit sortiert. Damit erhält der Pfleger einen kompletten Fütterungsplan.

Eine weitere Anforderung ist die Bestellliste. Sie soll die Differenz des Maximalwertes und des aktuellen Lagerbestandes der jeweiligen Futtersorte anzeigen. Dazu wird die Anzahl der Einheiten von den maximalen Einheiten subtrahiert und als Kaufmenge ausgegeben.

```
SELECT `lager`.`Sorte`, `lager`.`Max Einheit`-`lager`.`Anzahl Einheiten` AS 'Kaufmenge'
FROM `lager`
```

Abbildung 9: SQL-Bestellliste

Für den Tierpfleger ist außerdem eine Infoliste gefordert die alle Tiere eine Art auflistet und dabei alle Informationen zu dem Tier zur Verfügung stellt. Um eine Art auszuwählen soll diese in einem Abfragefenster eingegeben werden. Diese Abfrage würde in PHP umgesetzt werden und überschreitet damit den Rahmen, der für dieses Projekt vorgesehen ist. Stellvertretend für die Eingabe, die in PHP getätigt wurde wird hier Affe eingesetzt um das Testen des Codes in SQL zu ermöglichen.

```
SELECT `tier`.`Tierart`, `tier`.`Name`, `tier`.`Geschlecht`, `tier`.`Geburtsdatum`, `fuetterung`.`Sorte`, `fuetterung`.`Futtermenge (kg)`
FROM `fuetterung`
    , `tier`
    , `lager`
WHERE `tier`.`Tierart` = 'Affe'
AND `fuetterung`.`Sorte` = `lager`.`Sorte`
AND `fuetterung`.`Tier_ID` = `tier`.`Tier_ID`
```

Abbildung 10: SQL-Infoliste

Ausblick

Funktionsweise des Lagerbestandes

Die Veränderung des Lagerbestandes soll über ein Barcode System im Zoo verändert werden. Dabei werden Einheiten die hinzugefügt werden eingescannt und mittels entsprechendem SQL Code in die Datenbank eingefügt. Auch Futter welches von dem Tierpfleger zur Fütterung entnommen wird soll mittels Waage gemessen, und entsprechend in der Datenbank eingepflegt werden.

Nutzeroberfläche

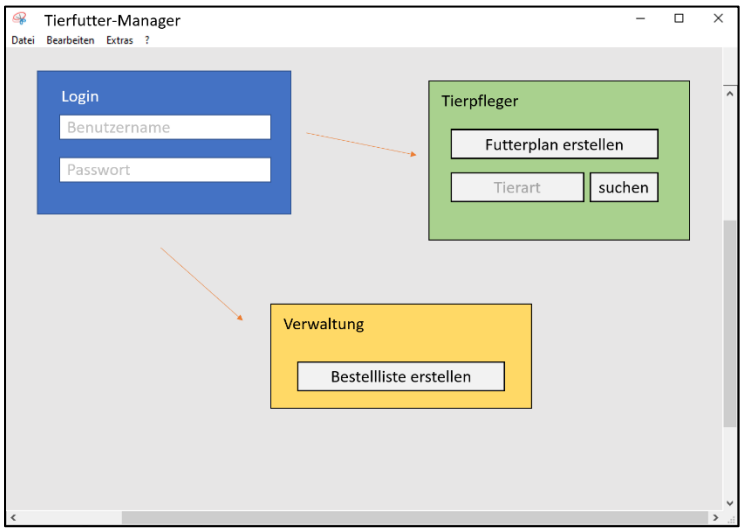

Abbildung 11: Nutzeroberfläche - Skizze

Die Nutzeroberfläche soll den Zugriff auf die Datenbank ermöglichen. Dabei gibt es zwei User-Gruppen die jeweils unterschiedliche Optionen haben. Die Tierpfleger haben die Möglichkeit die Fütterungspläne und Infolisten zu den Tieren zu erstellen. Die Verwaltung hat Zugriff auf die Bestellliste um entsprechendes Futter nachzubestellen. Das erstellen einer Website mit Nutzeroberfläche soll in diesem Projekt nicht umgesetzt werden. Um jedoch zu zeigen wie die Anwendung aussehen soll sind ist die Nutzeroberfläche in Abbildung 11 skizziert.

Testing

Anhand der aufgestellten Anforderungen an die Software sind im weiteren Verlauf die aufgezeigten Testfälle erstellt worden. Die Teste sind unterschieden in Unittest, und Systemtest, sowie durch eine finalen Abnahmetest. Dabei sollen die Teste in die Systeme des Tierpflegers und der Verwaltung unterschieden werden. Für jedes System gibt es unterschiedliche Anforderungen, wie sie bereits im Vorfeld aufgeführt sind.

Die Teste sollen dabei in einer Testumgebung getestet werden. Dabei werden nur die einzelnen Anforderungen überprüft und getestet. Um die Test erfolgreich zu bestehen werden einzelne Unitteste abgearbeitet, welche in Systemtests danach überprüft werden. Dabei sollen die einzelne Anforderungen unterschieden werden. Ein Beispiel für einen Unittest in unserem Projekt ist, dass der Button für die Suche nach Tieren funktionieren soll. Dazu muss eine Tierart in die Suche eingeschrieben werden und die Suche muss beginnen. Der Test ist erfolgreich abgeschlossen, wenn der Suchvorgang beginnt. Die einzelnen Teste sind in den Testfällen als Unittest sowie als Systemtest anzusehen.

Nach erfolgreichen Unittests werden Systemtests für die einzelnen Anforderungen durchgeführt. Hierbei soll Systemübergreifend mehrere Unittests abgearbeitet werden und ohne Fehler am besten durchlaufen. Dazu gehört zum einen, dass das Login möglich ist und zum Beispiel ein Futterpaln sich erstellen lässt. Wobei hier die richtigen Werte berücksichtigt werden müssen, so wie si in der Datenbank implementiert sind.

Im Abnahmetest sollen alle Anforderungen hintereinander getestet werden und aus Kundensicht beurteilt werden. Hierbei sollen keine Fehler mehr auftreten und die einzelnen Systeme reibungslos funktionieren. Hierbei müssen alle Anforderungen, die an unsere System gestellt sind, vorher getestet sein.

Insgesamt gibt es für unser Projekt im System des Tierpflegers 6 Testfälle. Benannt sind diese Testfälle von TF-TP-01 bis TF-TP-06. Beim System für den Verwalter sind 5 Testfälle aufgeführt. Diese sind von TF-VW-01 bis TF-VW-05 beschrieben. Alle Testfälle sind im Anhang aufgeführt, werden aber im Folgenden einzeln in Kurzform beschrieben.

Der erste Testfall für den Tierpfleger ist der TF-TP-01. In diesem Testfall soll das Login am Programm getestet werden. Bedingung hierfür ist, dass der Benutzer ein gültiges Passwort und Benutzernamen besitzt. Diese sind individuell und sollen mit mehreren unterschiedlichen Parametern einsetzbar, sodass dieser Test öfters wiederholt werden muss. Im Einzelnen gibt es 7 Unitteste, die in diesem Fall abgearbeitete werden müssen. Ohne erzielen eines erfolgreichen Test kann das System nicht arbeiten.

Beginnen muss es damit, dass die Software geöffnet werden muss und hochfahren muss, sodass der Startbildschirm der Anwendung erscheint. Dabei muss auf der Benutzeroberfläche die Möglichkeit des Logins erscheinen. Dieses soll in Form einer Anmeldefeldes erscheinen. In dem Anmeldefeldes muss es die Möglichkeit geben, dass ein Benutzername und ein Passwort, welches gültig und freigeschaltet ist, eingeschrieben werden kann. Nach Eingabe von gültigen Login-Daten müssen diese gesendet werden, damit der Benutzer am System angemeldet werden kann.. Erfolgreich ist der Test, wenn alle Möglichkeiten durchgetestet sind und der Benutzer angemeldet ist. Daraufhin wird da Abmelden des Benutzer gleich mitgetestet. Daraus resultiert, dass der Benutzer sich erfolgreich An- und Abmelden kann an unsere Software.

Der TF-TP-02 ist das Gegenteil vom ersten Testfall TF-TP-01. Dabei soll geprüft werden, dass mit einem nicht autorisiertem Benutzernamen und/oder Passwort, die Anmeldung verhindert werden soll. Der Test ist erfolgreich, wenn bei Falscheingabe eine Fehlermeldung auftaucht und keine Anmeldung an der Software durchgeführt wird. Diese beiden Testfälle werden bei der Verwaltung genauso benötigt. Dort werden sie mit den Testfall-IDs TF-VW-01 und TF-VW-02 behandelt und abgearbeitet. Unterschieden müssen die beiden Logins, da die Verwaltung und der Tierpfleger unterschiedliche Rechte an der Software besitzen.

Um eine weitere Anforderung des Tierpflegers abzudecken, ist der Testfall TF-TP-03 wichtig. Daher auch die Priorität hoch hierbei. Es soll laut Anforderung eine Tierliste generiert werden, die durch die Suche nach Tierarten genutzt werden soll. Sortiert soll diese alphabetisch nach den Namen der Tiere. Hierfür ist der Testfall TF-TP-01 erfolgreich abzuschließen. In der Benutzeroberfläche muss es eine Bereich für die Erstellung dieser Liste geben. In diesem muss es die Möglichkeit einer Eingabe geben, um eine Tierart einzutippen. Diese Tierart muss in der Tabelle Tier in der Datenbank aufgelistet sein, sodass die Suche erfolgreich sein kann. Dabei muss die Datenbank noch nicht die richtigen Tiere unseres Zoos besitzen. Wenn die Anzeige der Tierart erfolgreich ist, sollen weitere Elemente angezeigt werden. Dafür soll die Datenbank immer weiter gefüllt werden, aber so, dass nur ein Element immer neu ist. Weitere Ausgaben sollen dabei sein, Name, Tier-ID, Geburtsdatum als Alter, welcher einen extra Testfall besitzt, Geschlecht. Diese stehen in der Tabelle des Tieres. Neben diesen Elementen soll aus der Tabelle fuetterung die Fütterungsmenge angezeigt werden, sodass der Tierpfleger einen Überblick über die Menge des vorzubereitenden Futters bekommt. Diese Tests müssen erst alleine erfolgreich abgeschlossen sein und dann im System getestet werden, damit eine komplette Liste mit den richtigen Einträgen angezeigt wird. Erfolgreich ist dieser Test, wenn am Ende eine richtige Liste mit den richtigen Elementen erscheint.

Der nächste Testfall beinhaltet, dass der Tierpfleger keine Bearbeitungsmöglichkeiten in der Software hat. Diese soll ausschließlich die Verwaltung verwalten. Das bedeutet der TF-TP-04 beinhaltet, dass es verhindert werden muss, wenn der Tierpfleger angemeldet ist, die Tabellen und Datenbanken aufzurufen und/oder keine Änderungen, Speicherungen oder Löschungen an der Software vorzunehmen. Dazu benötigt die Software eine Schaltfläche, in der ein Bearbeitungsfeld mit hinzufügen, löschen und bearbeiten erscheint. Diese soll für den Tierpfleger verhindert werden. Der Test ist erfolgreich, wenn diese Anwendung nicht gestartet wird. Sollte es trotzdem starten, sollte der

Tierpfleger keine Möglichkeiten haben, die Daten zu bearbeiten. Dazu soll er eine Fehlermeldung bekommen, dass die Berechtigung fehlt, um Daten zu ändern, löschen oder zu speichern.

Ein weiterer Testfall ist der TF-TP-05. Hierbei soll getestet werden, ob ein Fütterungsplan erstellt werden kann. Dazu benötigt die Benutzeroberfläche eine Anwendung, dass so ein Plan erstellt werden kann. Der Plan muss mit einem Soll-Tag → Ist-Tag Vergleich geprüft werden kann, damit der richtige Tag erscheint. Der Plan soll nach Wochentag und Fütterungszeit sortiert sein. Dieser muss aus der Tabelle Tier die Tierart, den Namen und Tier-ID besitzen, sowie aus der Tabelle fuetterung die Futtermenge und Fütterungszeit. Nach dieser Zeit soll der Plan erstellt sein. Diese sollen alleine getestet werden und dann als System getestet werden. Der Test ist erfolgreich, wenn alle Daten zu den Eintragungen in der Tabelle passen. Dazu reichen Testdaten in der Tabelle, um die Liste zu generieren. Einzige wichtige dabei ist, dass die Zeiten korrekt sein müssen. Zum Beispiel muss die Uhrzeit als 11:25 Uhr angegeben werden.

Letzter Testfall ist der TF-TP-06, welcher den Test der Umwandlung des Geburtsdatum in Jahre prüfen soll, damit das Alter angezeigt wird. Hierbei muss es einen Soll-Ist-Vergleich zwischen Geburtsdatum und aktuellen Datum geben.

Bei der Verwaltung gibt es wie bei dem Tierpfleger zwei Testfälle, die für das Login und das Verhindern des Logins geschrieben sind. Diese werden vergleichbar ausgeführt und nicht weiter detailliert aufgeführt.

Damit ist der Testfall TF-VW-03 der nächste Testfall, der betrachtet wird. Hierbei soll die Bearbeitung der Daten für die Verwaltung geprüft werden. Dabei soll die Bearbeitungsfunktion, Löschfunktion und Hinzufügefunktion getestet werden. Erst einzelne Unitteste und dann im Systemtest geprüft. Dabei sollen alle Einträge für Tiere, Fütterung und Lager bearbeitbar, löschbar und hinzufügbar sein. Wichtig dabei ist die Rechteauthentifizierung, sodass das System erkennt, dass die Verwaltung daran arbeitet und kein Tierpfleger. Der Test ist erfolgreich, wenn die geänderten Daten gespeichert sind und bei dem Tierpfleger korrekt angezeigt werden.. Dazu ist übergreifend der TF-TP-03 und TF-TP-05 zu beachten. Diese müssen automatisch erkennen, dass diese Daten geändert worden sind.

Der TF-VW-04 ist ein Testfall zur Regenerierung einer Einkaufsliste für die Verwaltung. Dabei muss es auf der Benutzerfläche einen Button zur Erzeugung einer Einkaufsliste geben. Damit startet der Test. Getestet werden muss, ob der maximale Wert des Futters aus der Tabelle des Lagers angezeigt wird. Danach muss abgefragt werden, ob der aktuelle Wert des Futters im Lager angezeigt werden kann. Daraus muss die Differenz erzeugt werden, welche auf Richtigkeit getestet werden muss. Dieses muss innerhalb einer Testumgebung erzeugt werden. Wenn dieses Ergebnis erfolgreich getestet ist, soll es eine Liste mit der dazugehörigen Sorte ausgeben. Erfolgreich ist der Test, wenn eine Liste und Differenz zwischen maximalen Lagerbestand und aktuellem Lagerbestand ist.

Der vorerst letzte Testfall ist der TF-VW-05. Hierbei soll getestet werden, dass ein Warnhinweis bei zu niedrigem Lagerbestand der Verwaltung erscheint. Dazu muss in der Datenbank ein Minimalwert hinterlegt sein. Dieser muss mit dem aktuellen Stand des Lagers kommunizieren. Dazu muss die Kommunikation geprüft werden und auf Richtigkeit überprüft sein. Bei Unterschreitung des Minimalwertes wird ein Alarmsignal erscheinen müssen. Erfolgreich sind die Teste, wenn bei zu niedrigem Stand, ein Warnsignal eerscheint.

Abschließend muss ein Abnahmetest in Form von Kunden durchgeführt werden. Dazu müssen die Datenbanken und Tabellen, sowie die richtigen Einträgen vorgenommen werden. Ohne diesen Test lässt sich nicht garantieren, dass die Software ohne Probleme arbeitet und die richtigen Daten beansprucht.

Projektname: Tierfutter Mini-Zoo	
Testfall: Tierpfleger	
Testfall-ID: TF-TP-01	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): mittel	Designed Datum: 28.05.2020
Modulname: Anmeldung System	Durchgeführt von:
Testtitel: Testen der Login-Funktion am Programm Tierfutter	Durchführungsdatum:
Beschreibung: Überprüfen der Anmeldung am Programm mit einem gültigem Benutzernamen und Passwort	

Voraussetzung: Der Benutzer (Tierpfleger) hat einen gültigem Benutzernamen und gültiges Passwort, welche zur Berechtigung zum Arbeiten am Programm führen

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Öffnen der Software		Software soll geöffnet werden			
2	Aufzeigen des Anmeldefeldes		Anmelden des Benutzer sollte ermöglicht werden			
3	Eingeben eines gültigem Benutzernamen	Beispiel: Zooname	Benutzername kann eingegeben werden			
4	Eingeben eines gültigen Passwortes	Beispiel: Zooname	Passwort kann eingegeben werden			
5	Senden der Anmeldedaten		Benutzer ist angemeldet am System			
6	Abmelden des Benutzers	Über Button abmelden	Benutzer ist erfolgreich von System getrennt			
7	Neues Aufzeigen des Anmeldefeldes		Benutzer kann sich neu anmelden			
8	Ende Test					

Projektname: Tierfutter Mini-Zoo	
Testfall: Tierpfleger	
Testfall-ID: TF-TP-02	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): mittel	Designed Datum: 28.05.2020
Modulname: Anmeldung System	Durchgeführt von:
Testtitel: Testen der Login-Funktion am Programm Tierfutter mit falschem Login-Daten	Durchführungsdatum:
Beschreibung: Überprüfen der Anmeldung am Programm mit einem ungültigem Benutzernamen und Passwort, welche das Login nicht durchführen lassen sollen	

Voraussetzung: Der Benutzer (Tierpfleger) hat ein ungültigem Benutzernamen und/oder gültiges Passwort eingeben

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Öffnen der Software		Software soll geöffnet werden			
2	Aufzeigen des Anmeldefeldes		Anmelden des Benutzer sollte ermöglicht werden			
3	Eingeben eines ungültigen Benutzernamen	Beispiel: Zooname falsch, keine Berechtigung	Benutzername kann eingegeben werden			
4	Eingeben eines ungültigen Passwortes	Beispiel: Zooname falsch, keine Berechtigung	Passwort kann eingegeben werden			
5	Senden der Anmeldedaten		Benutzer darf nicht an System angemeldet werden. Anzeigen Fehler Eingabedaten			
6	Abmelden des Benutzers	automatisch	Keine Anmeldung des Benutzers			
7	Neues Aufzeigen des Anmeldefeldes		Benutzer kann sich neu anmelden			
8	Ende Test					

Projektname: Tierfutter Mini-Zoo	
Testfall: Tierpfleger	
Testfall-ID: TF-TP-03	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): hoch	Designed Datum: 28.05.2020
Modulname: Tierliste generieren	Durchgeführt von:
Testtitel: Testen der Suchfunktion	Durchführungsdatum:
Beschreibung: Überprüfung der Suchfunktion nach Tierarten	

Voraussetzung: TF-TP-01 erfolgreich. Die Software wird ausgeführt.

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Öffnen der Suchfunktion		Aufzeigen eines Eingabefeldes mit Suchfunktion			
2	Eingabe einer Tierart, die im System vorhanden und aufgeführt ist	`Tierart` varchar(50) Beispiel: Tiger	Anzeige der Tierart in einer Liste sortiert nach Tier-Name			
3	Weitere Anzeigeelemente in der Liste anzeigen	Table tier: `Tier-ID` int(11); `Name` varchar(50); `Geburtsdatum` date; `Geschlecht` varchar(20) table fuetterung: `Fuetterung-smenge` varchar(20)	Nach Suche der Tierart soll die Liste neben der Tierart, die weiteren Elemente: Tier-ID, Name, Geburtsdatum, Geschlecht, Fuetterungsmenge aufzeigen			
4	Ende Taste		Aufzeigen des Eingabefeldes für Suchfunktion			
5	Beginn neue Suche		Eingabe eines falschen/ nicht vorhandenen Tierart			
6	Eingabe falscher Tierart		Keine Anzeige von Tieren. Fehlerhinweis: nicht aufgeführt in der Liste			
7	Ende Test		Aufzeigen de Suchfunktion			

Projektname: Tierfutter Mini-Zoo	
Testfall: Tierpfleger	
Testfall-ID: TF-TP-04	Entworfen von: Team Tierfutter Team Tierfutter
Priorität (hoch, mittel, tief): hoch	Designed Datum: 28.05.2020
Modulname: Bearbeitung Datenbanken	Durchgeführt von:
Testtitel: Eingabeerlaubnis von Daten für Tiere vom Tierpfleger nicht erlauben	Durchführungsdatum:
Beschreibung: Der Tierpfleger darf keine Daten in den Datenbänken bearbeiten, speichern oder löschen dürfen.	

Voraussetzung: TF-TP-01 erfolgreich. Datenbänke bearbeitbar und veränderbar

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Datenbank aufrufen		Die Datenbank Tier soll aufgerufen werden			
2	Benutzer möchte Datenbank bearbeiten		Rechteauthentifizie-rung abfragen			
3	Benutzer probiert Änderung an der Datenbank		Eingabe veränderten Daten in der Datenbank beginnt und soll verhindert werden Fehlermeldung: Berechtigung fehlt			
4	Eingegebene Daten auf Datenbank nicht speicherbar		Der Tierpfleger kann bearbeitete Daten nicht speichern. Fehlermeldung: Berechtigung fehlt			
5	Eingegebene Daten in der Datenbank dürfen nicht gelöscht werden		Löschen nicht möglich Fehlermeldung: Berechtigung fehlt			
6	Erfolgreicher Test bei verhindern aller Eingaben		Der Test ist erfolgreich, wenn alle Daten erfolgreich nicht verändert werden können			
7	Test Ende					

Projektname: Tierfutter Mini-Zoo	
Testfall: Tierpfleger	
Testfall-ID: TF-TP-05	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): hoch	Designed Datum: 28.05.2020
Modulname: Fütterungsplan erstellen	Durchgeführt von:
Testtitel: Anzeigen Fütterungsplan	Durchführungsdatum:
Beschreibung: Überprüfung der Anzeige eines Futterplans für einen Wochentag	

Voraussetzung: TF-TP-01 erfolgreich, Datenbank erfolgreich

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Erstellung über einen Button		Nach klicken auf den Button soll ein Futterplan erstellt werden.			
2	Futterplan auf Datum und Wochentag prüfen		Der Futterplan soll das heutige Datum und heutigen Wochentag prüfen..			
3	Erstellung Fütterungsplan	Table tier: `Tierart` varchar(50); `Name` varchar(50); `Tier-ID` int(11); Table fuetterung: `Futtermenge` tint(11); `Fuetterungs-zeit` time	Ein Futterplan soll nach der Fütterungszeit und Wochentag sortiert aufgezeigt werden.			
4	Überprüfung auf richtigen Wochentag	Table fuetterung: `Fuetterungstag` Varchar(20); Realtime date	Anzeigen des Futterplan fürs richtige Datum/Tag			
5	Sortierung der Liste nach Zeit für alle Fütterungen	Table fuetterung: `fuetterungs-zeit` time	Die Liste soll nach Uhrzeiten sortiert ausgegeben werden für den jeweiligen Wochentag			

Projektname: Tierfutter Mini-Zoo	
Testfall: Tierpfleger	
Testfall-ID: TF-TP-06	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): tief	Designed Datum: 28.05.2020
Modulname: Alter verwalten	Durchgeführt von:
Testtitel: Anzeigen des Alters durch Geburtsdatum	Durchführungsdatum:
Beschreibung: Überprüfung des Alters, welches durch das Geburtsdatum ermittelt wird	

Voraussetzung: TF-TP-01/TF-VW-01 erfolgreich, Zugriff Datenbank

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Anlegen eines Datums		Datum in Datenbank gespeichert			
2	Erzeugen einer Liste nach Tierart	Siehe TF-TP-03	Anzeige des Alters als Jahre			
3	Ende Test		Anzeigen des richtigen Alters			

Projektname: Tierfutter Mini-Zoo	
Testfall: Verwaltung	
Testfall-ID: TF-VW-01	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): mittel	Designed Datum: 28.05.202028.05.2020
Modulname: Anmeldung System	Durchgeführt von:
Testtitel: Testen der Login-Funktion am Programm Tierfutter	Durchführungsdatum:
Beschreibung: Überprüfen der Anmeldung am Programm mit einem gültigem Benutzernamen und Passwort	

Voraussetzung: Der Benutzer (Tierpfleger) hat einen gültigem Benutzernamen und gültiges Passwort, welche zur Berechtigung zum Arbeiten am Programm führen

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Öffnen der Software		Software soll geöffnet werden			
2	Aufzeigen des Anmeldefeldes		Anmelden des Benutzer sollte ermöglicht werden			
3	Eingeben eines gültigem Benutzernamen	Beispiel: Zooname	Benutzername kann eingegeben werden			
4	Eingeben eines gültigen Passwortes	Beispiel: Zooname	Passwort kann eingegeben werden			
5	Senden der Anmeldedaten		Benutzer ist angemeldet am System			
6	Abmelden des Benutzers	Über Button abmelden	Benutzer ist erfolgreich von System getrennt			
7	Ende Test					

Projektname: Tierfutter Mini-Zoo	
Testfall: Verwaltung	
Testfall-ID: TF-VW-02	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): mittel	Designed Datum: 28.05.2020
Modulname: Anmeldung System	Durchgeführt von:
Testtitel: Testen der Login-Funktion am Programm Tierfutter mit falschem Login-Daten	Durchführungsdatum:
Beschreibung: Überprüfen der Anmeldung am Programm mit einem ungültigem Benutzernamen und Passwort, welche das Login nicht durchführen lassen sollen	

Voraussetzung: Der Benutzer (Tierpfleger) hat ein ungültigem Benutzernamen und/oder gültiges Passwort eingeben

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Öffnen der Software		Software soll geöffnet werden			
2	Aufzeigen des Anmeldefeldes		Anmelden des Benutzer sollte ermöglicht werden			
3	Eingeben eines ungültigen Benutzernamen	Beispiel: Zooname falsch, keine Berechtigung	Benutzername kann eingegeben werden			
4	Eingeben eines ungültigen Passwortes	Beispiel: Zooname falsch, keine Berechtigung	Passwort kann eingegeben werden			
5	Senden der Anmeldedaten		Benutzer darf nicht an System angemeldet werden. Anzeigen Fehler Eingabedaten			
6	Abmelden des Benutzers	automatisch	Keine Anmeldung des Benutzers			
7	Neues Aufzeigen des Anmeldefeldes		Benutzer kann sich neu anmelden			
8	Ende Test					

Projektname: Tierfutter Mini-Zoo	
Testfall: Verwaltung	
Testfall-ID: TF-VW-03	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): tief	Designed Datum: 28.05.2020
Modulname: Bearbeitung Datenbanken Tier	Durchgeführt von:
Testtitel: Bearbeiten der Datenbanken	Durchführungsdatum:
Beschreibung: Prüfen der Bearbeitungsfunktion der Verwaltung in den Datenbanken	

Voraussetzung: TF-VW-01; TF-TP-03, Berechtigung zum bearbeiten

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Datenbanken aufrufen		Die Datenbank Tier, Datenbank Lager und Datenbank Fütterung soll aufrufbar sein			
2	Benutzer möchte Datenbank bearbeiten		Rechteauthenti-fizierung abfragen und genehmigen			
3	Datenbank Tier aufrufen		Anzeigen der Datenbank Tier			
4	Daten in der Datenbank Tier bearbeiten	Table `Tier`	Änderungen in der Datenbank erkennbar und in der Liste; Testen in der TF-TP-03			
5	Daten in der Datenbank Tier löschen	Table `Tier`	Gelöschter Eintrag nicht mehr in der Suchfunktion vorhanden; Testen in der TF-TP-03			
6	Daten in der Datenbank Tier hinzufügen	Table `Tier`	Hinzugefügter Eintrag in der Liste und Datenbank erkennbar; Testen in der TF-TP-03			
7	Für jede Datenbank durchführen	Table lager und fuetterung	Die Datenbanken sollen innerhalb der			
8	Ende Test		Ende Test ist erfolgreich, wenn in allen Datenbanken erfolgreich Daten geändert werden können und diese korrekt angezeigt werden			

Projektname: Tierfutter Mini-Zoo	
Testfall: Verwaltung	
Testfall-ID: TF-VW-04	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): hoch	Designed Datum: 28.05.2020
Modulname: Einkaufsliste regenerieren	Durchgeführt von:
Testtitel: Testen der Erstellung einer Einkaufsliste	Durchführungsdatum:
Beschreibung: Überprüfen ob eine Einkaufsliste erstellt wird, wenn die Verwaltung danach fragt	

Voraussetzung: TF-VW-01

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Anzeigen des Maximalwertes aus der Datenbank Lager	`Max Einheit`	Anzeige des Maximalwertes, welcher durch die Verwaltung vorgebenen ist (Maximaler Lagerplatz)			
2	Anzeigen der aktuellen Anzahl im Lager	Ànzahl Einheiten`	Anzeigen des aktuellen Lagerbestandes. Der aktuelle Lagerbestand wird aus der der benötigten Futtermenge pro Tag ständig aktualisiert.			
3	Differenz anzeigen	`Max Einheit` Minus Ànzahl Einheiten`	Anzeigen der Differenz zwischen den Punkt 1 und Punkt 2			
4	Liste erstellen für benötigte Einkaufsmenge	Table `lager` `sorte` varchar(50)	Die Liste soll die Differenz aus Punkt 3 wiedergeben. Dazu soll die Futtersorte ausgegeben werden			
5	Liste anzeigen		Die fertige Liste soll angezeigt werden			

Projektname: Tierfutter Mini-Zoo	
Testfall: Verwaltung	
Testfall-ID: TF-VW-05	Entworfen von: Team Tierfutter
Priorität (hoch, mittel, tief): hoch	Designed Datum: 28.05.2020
Modulname: Benachrichtigung Minimalwert	Durchgeführt von:
Testtitel: Testen Minimalwert	Durchführungsdatum:
Beschreibung: Bei der Unterschreitung des Minimalwertes zum Istwert, soll eine Warnmeldung erscheinen	

Voraussetzung: TF-VW-01

Nr.	Testschritte	Testdaten	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status (i.O/n.i.O)	Anmerkung
1	Minimalwert in der Datenbank aufgelistet	Min. Einheit int(11)	Die richtigen Minimalwerte müssen bei der Implementierung der Software richtig eingetragen sein			
2	Kommunikation mit Ist-Wert des Lagerbestandes		Minimalwert muss mit dem Ist-Wert des Lagerbestandes verbunden und verglichen werden			
3	Unterschreitung Minimalwert		Anzeigen eines Warnhinweises für die Verwaltung mit der Möglichkeit der sofort zu bestellen			
4	Ende Test		Erfolgreicher Test mit Anzeige des Warnhinweises			