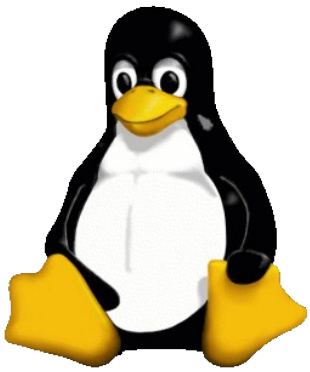
Entwicklung eines

Kundenverwaltungssystem

für eine Webhosting-Firma



Projektarbeit Datenbanken

NTA FH Isny
12. Info

David Mayr

Inhaltsverzeichnis

Aufgabenstellung

Eine Webhosting-Firma möchte ihre Kunden- und Auftragsverwaltung mit einem erweiterbarem Datenbank-gestützten System abwickeln.

Es gilt Kunden, deren Verträge und Rechnungen, sowie die zu jedem Vertrag gehörenden Benutzer-Konten (mit Mailbox, Domain und evtl. einer Datenbank) zu verwalten.

Lastenheft

1 Zielbestimmung

Mit der Software sollen Kunden- und deren Vertrags-Daten (gebuchte Webhosting-Leistungen - Domains, Mailboxen, Datenbanken) von einer zuständigen Person erfasst und modifiziert werden können. Dabei sollen die Daten möglichst schnell und einfach geändert – aber auch möglichst übersichtlich dargestellt werden können.

Alle Daten der Anwendung sollen in einer relationalen Datenbank gespeichert werden.

Da die Software vorausichtlich nur auf einem PC laufen wird, sollen die Datenbank-Zugangsdaten in einer Konfigurationsdatei abgespeichert werden können.

Es soll eine Erinnerungsfunktion realisiert werden, die zur manuellen Erstellung von Rechnungen auffordert, sobald es für einen Vertrag seit über 12 Monaten keine Rechnung gab. Ausserdem soll auf einen Blick der Stand offener Rechnungen ersichtlich sein.

2 Produkteinsatz

Zielgruppe: Verwaltung einer Webhosting-Firma

3 Produktfunktionen

LF10:	**Kunde**	anlegen, ändern und löschen
LF20:	**Vertrag**	anlegen, ändern und löschen
LF30:	**Rechnung**	anlegen, ändern und löschen + an Rechnungserstellung erinnern
LF40:	**Benutzer**	anlegen, ändern und löschen
LF50:	**Domain**	anlegen, ändern und löschen
LF60:	**Mailbox**	anlegen, ändern und löschen
LF70:	**Datenbank**	anlegen, ändern und löschen

4 Produktdaten

LD10: **Kunde**
id, Firma, Anrede, Name, Vorname, Strasse, HausNr, PLZ, Ort, Telefon, Fax, Email, erstelltAm

LD20: **Vertrag**
id, Kunde, Datum, gekündigt?, gekündigtAm

LD30: **Rechnung**
id, Vertrag, Datum, BetragNetto, bezahlt?, bezahltAm

LD40: **Benutzer**
Name, Vertrag, uid, Passwort, Speicher, Mailbox, Domain, Datenbank

LD50: **Domain**
Name, hatMailinglisten?

LD60: **Mailbox**
Name, Speicher, Adresse

LD70: **Datenbank**
Name, Passwort, Speicher

Pflichtenheft

1 Aufgaben

Verwaltung der Daten von Webhosting-Kunden.

2 Adressaten

Auftraggeber: LunaBOX Network Solutions
Auftragnehmer: D. Mayr

3 Inhalt / Produktfunktionen / Produktdaten

Die Software, im weiteren SW bezeichnet, soll folgende Merkmale besitzen:

Kunde
- Daten:
  - **id**, Firma, Anrede, Name, Vorname, Strasse, HausNr, PLZ, Ort, Telefon, Fax, Email, erstelltAm
- Funktionen:
  - anlegen, ändern, löschen

Vertrag
- Daten:
  - **id**, Kunde, Datum, gekündigt?, gekündigtAm
- Funktionen:
  - anlegen, ändern, löschen

Rechnung
- Daten:
  - **id**, Vertrag, Datum, BetragNetto, bezahlt?, bezahltAm
- Funktionen:
  - anlegen, ändern, löschen, an Erstellung erinnern

Benutzer
- Daten:
  - **Name**, Vertrag, uid, Passwort, Speicher, Mailbox, Domain, Datenbank
- Funktionen:
  - anlegen, ändern, löschen

Domain
- Daten:
  - **Name**, hatMailinglisten?
- Funktionen:
  - anlegen, ändern, löschen

Mailbox
- Daten:
  - **Name**, Speicher, Adresse
- Funktionen:
  - anlegen, ändern, löschen

Datenbank
- Daten:
  - **Name**, Passwort, Speicher
- Funktionen:
  - anlegen, ändern, löschen

4 Produktumgebung

Das Produkt soll auf einem Arbeitsplatz mit grafischer Oberfläche laufen.

Als stabile Basis soll das freie Betriebssystem Linux verwendet werden.

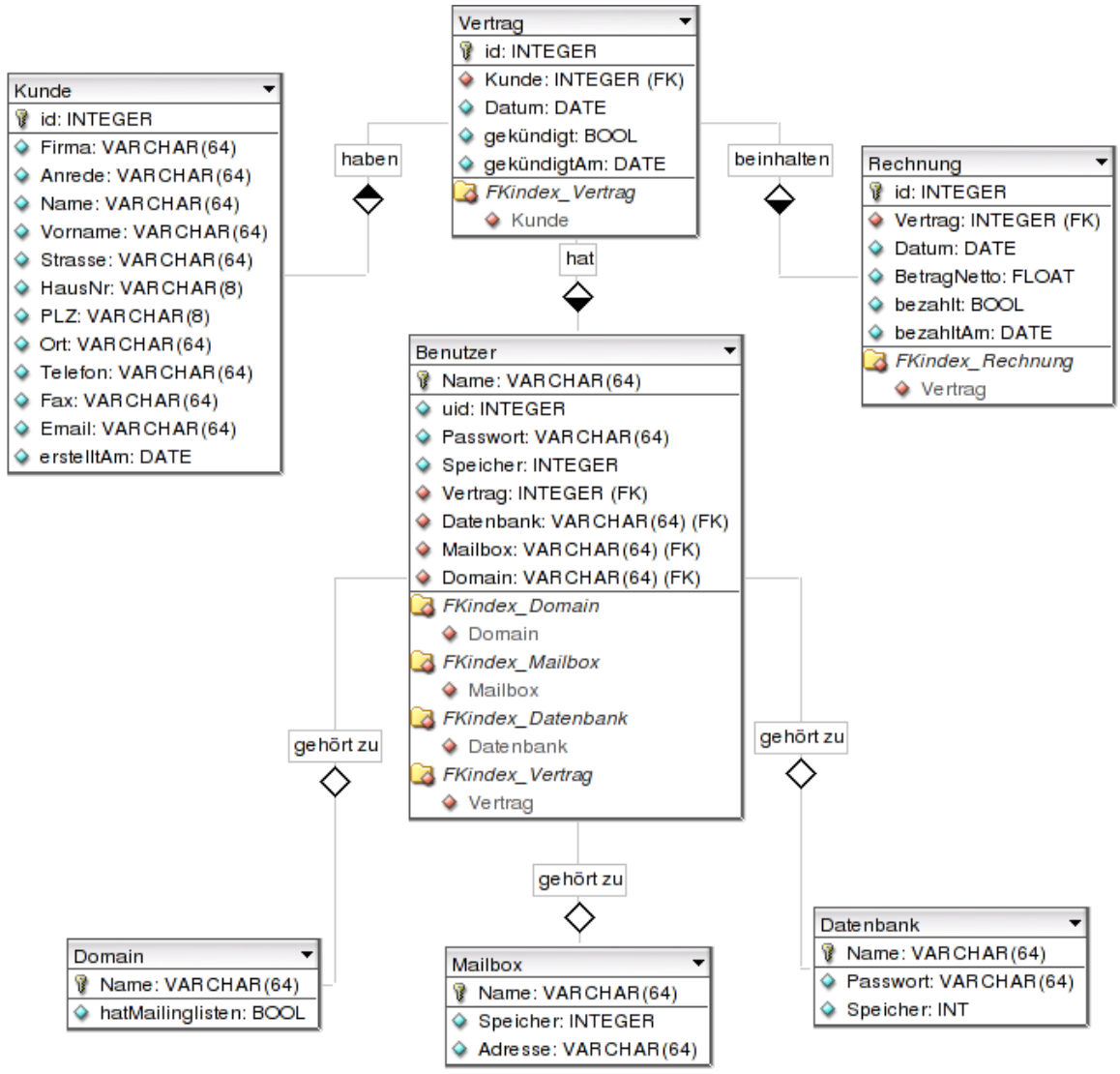
Um trotz der Festlegung des Betriebssystems möglichst portabel zu bleiben, wurde als GUI-Bibliothek "QT"[1] von der Firma Trolltech verwendet.

QT ist für die Nutzung in nicht-kommerziellen Anwendungen frei – prominentestes Beispiel ist die unter Linux weit verbreitete Desktop-Umgebung KDE.

1 http://www.trolltech.com/ - gibt's für Linux, Windows & Mac ... und sogar für PDAs und Handys!

Konzeption der Datenbank

1 ER-Modell



- Jeder *Kunde* hat einen oder mehrere *Verträge*.
- Zu jedem *Vertrag* gibt es meist mehrere *Rechnungen*.
- Zu jedem *Vertrag* gehört mindestens ein *Benutzer*.
- Jeder *Benutzer* muss genau eine *Mailbox* haben.
- Jeder *Benutzer* kann eine oder keine *Domain* haben.
- Jeder *Benutzer* kann eine oder keine *Datenbank* haben.

Obiges Diagramm wurde mit dem freien Programm "DB-Designer" erstellt.

2 Datenbank-Tabellen

2.1 Kunde

Feldname	Datentyp	Beschreibung
id	int(10), primärschlüssel	Kundennummer
Anrede	varchar(64)	(sh. Feldname)
Firma	varchar(64)	(sh. Feldname)
Name	varchar(64)	(sh. Feldname)
Vorname	varchar(64)	(sh. Feldname)
Strasse	varchar(64)	(sh. Feldname)
HausNr	varchar(8)	(sh. Feldname)
PLZ	varchar(8)	(sh. Feldname)
Ort	varchar(64)	(sh. Feldname)
Telefon	varchar(64)	(sh. Feldname)
Fax	varchar(64)	(sh. Feldname)
eMail	varchar(64)	(sh. Feldname)
erstelltAm	date	Erstellungsdatum

2.2 Vertrag

Feldname	Datentyp	Beschreibung
id	int(10), primärschlüssel	Vertragsnummer
Kunde	int(10), fremdschlüssel	ID des Kunden, zu dem dieser Vertrag gehört
Datum	date	Datum des Vertragsabschlusses
gekündigt	tinyint(1) / bool	Wurde dieser Vertrag gekündigt
gekündigtAm	date	Falls: wann wurde dieser Vertrag gekündigt

2.3 Rechnung

Feldname	Datentyp	Beschreibung
id	int(10), primärschlüssel	Rechnungsnummer
Vertrag	int(10), fremdschlüssel	VertragsNr, zu dem diese Rechnung gehört
Datum	date	Rechnungsdatum
BetragNetto	float	Rechnungbetrag ohne Steuer
bezahlt	tinyint(1) / bool	Wurde diese Rechnung bereits bezahlt?
bezahltAm	date	Falls: wann wurde diese Rechnung bezahlt?

2.4 Benutzer

Feldname	Datentyp	Beschreibung
Name	varchar(64), primärschlüssel	Name des Benutzer-Kontos
UID	int(10)	Unix User-ID des Benutzers
Passwort	varchar(64)	Passwort des Benutzers
Speicher	int(10)	Speicherquota auf Dateisystemebene
Vertrag	int(10), fremdschlüssel	Vertrag, zu dem dieser Benutzer gehört
Mailbox	varchar(64), fremdschlüssel	Mailbox,die zu diesem Benutzer gehört
Domain	varchar(64), fremdschlüssel	Domain,die zu diesem Benutzer gehört
Datenbank	varchar(64), fremdschlüssel	DB,die zu diesem Benutzer gehört

2.5

2.6 Mailbox

Feldname	Datentyp	Beschreibung
Name	varchar(64), primärschlüssel	Name der Mailbox
Speicher	int(10)	Speicherquota der Mailbox
Adresse	varchar(64)	Haupt-eMailadresse der Mailbox

2.7

2.8 Domain

Feldname	Datentyp	Beschreibung
Name	varchar(64), primärschlüssel	Domain-Name
hatMailinglisten	tinyint(1)	sind für diese Domain ML aktiviert?

2.9

2.10 Datenbank

Feldname	Datentyp	Beschreibung
Name	varchar(64), primärschlüssel	Datenbank-Name (= DB-Username)
Passwort	varchar(64)	DB-Passwort
Speicher	int(10)	Maximale Grösse der Datenbank in MB

Realisierung der Anwendung

1 Entwicklungsumgebung

1.1 Linux / GCC

Als freie und stabile Basis für das Projekt wurde Linux als Betriebssystem und die GNU Compiler Collection (GCC) verwendet. Als Hilfsmittel kam ausserdem das verbreitete Werkzeug make zum Einsatz.

1.2 QT / QT-Designer

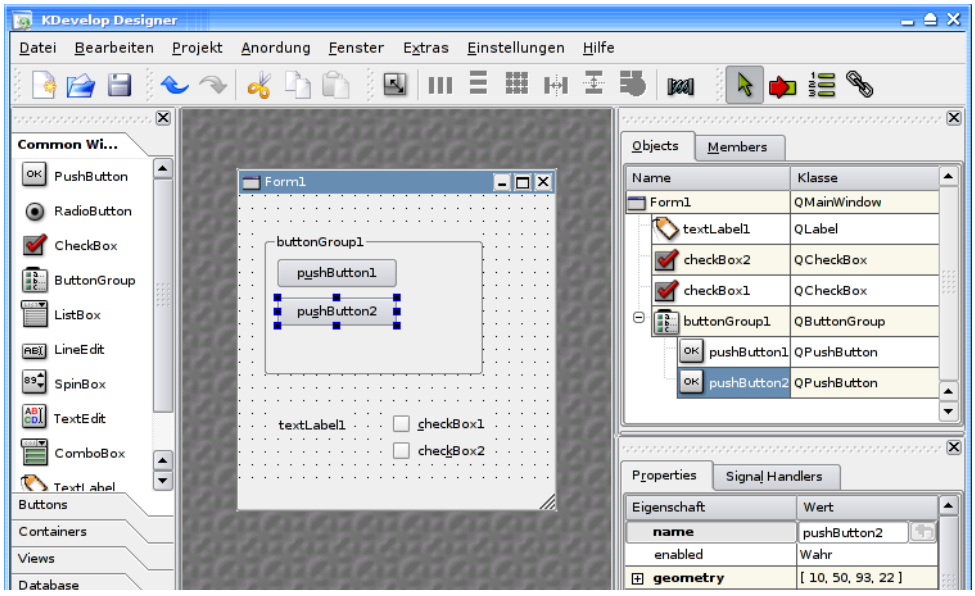
Die Norwegische Firma Trolltech bietet mit seiner Multi-Plattform GUI-Bibliothek eine gute Basis für graphische Benutzerinterfaces, das in einer speziellen Lizensierung angeboten wird: wird OpenSource-Software damit entwickelt, ist QT auch frei – sollen kommerzielle Produkte damit entwickelt werden, wird eine Lizenzgebühr für QT fällig.

Das besondere bei QT: der selbe Code läuft unter Linux, Windows und Mac. Dazu von der Trolltech-Homepage:

> *"Qt is a cross-platform C++ application framework developers can use to write single-source applications that run natively on Windows, Linux, Unix, Mac OS X and embedded Linux. Qt has been used to build thousands of successful commercial applications worldwide, and is the basis of the open source KDE desktop environment."*

Da KDE auf QT basiert ist auf einem Standard Linux-System in der Regel QT schon installiert. Zu QT gehört auch ein GUI-Designer, der QT-Designer.

Mit ihm lassen sich recht einfach die graphischen Elemente (Widgets) der Anwendung anordnen.



Der QT-Designer erzeugt XML-Dateien mit der Endung ".ui". Aus diesen GUI-Beschreibungsdateien werden später mit Hilfe von `qmake` und `make` automatisch die entsprechenden cpp- und Header-Dateien generiert.

Da zum Zeitpunkt der Erstellung dieser Projektarbeit gerade kürzlich QT in der Version 4 mit einigen Änderungen erschien, habe ich mich dazu entschieden weiterhin mit der bisherigen Version 3.4 zu arbeiten, unter anderem weil man dafür wesentlich mehr Informationen / Hilfe im Internet findet. Um für das Projekt explizit die ältere Version von QT zu verwenden, obwohl QT4 auch schon installiert war, ist es nötig gewesen den Pfad anzupassen:

```
export PATH=/usr/qt/3/bin/:$PATH
```

Ein weiterer Pluspunkt bei der Verwendung von QT ist, dass man sich nicht mit Makefiles auseinandersetzen muss – das mitgelieferte Werkzeug `qmake` erstellt mit `qmake -project && qmake` eine Makefile passend für die Dateien im aktuellen Arbeitsverzeichnis.

1.3 Eclipse mit CDT-Plugin

Da ich mit Eclipse als Java-Entwicklungsumgebung vertraut war, und vom CDT-Plugin, mit dem man auch C/C++ in Eclipse entwickeln kann, hörte, habe ich mich entschieden damit zu programmieren. Sowohl Eclipse als auch das Plugin sind frei. Ein Debugger wurde nicht verwendet.

1.4 MySQL

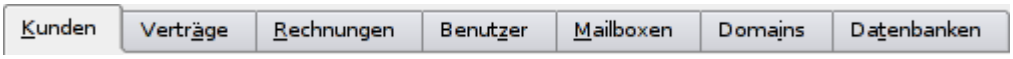
Als Datenbank wurde MySQL verwendet – einfach, bewährt und frei. Allerdings musste vom Standard-Tabellenformat "MyISAM" auf "InnoDB" umgestellt werden, damit Fremdschlüssel richtig funktionieren.

1.5 Subversion

Obwohl ich alleine an diesem Projekt gearbeitet habe, habe ich mich – wenn auch erst im Verlauf der Bearbeitung – entschlossen Subversion zur Versionskontrolle zu verwenden. Ich habe lokal einen Subversion-Server installiert und ein Repository "Kundenverwaltung" angelegt. So kann ich einfach die einzelnen Schritte bei der Entwicklung später nachvollziehen - und auch evtl. bei Bedarf zurückgehen.

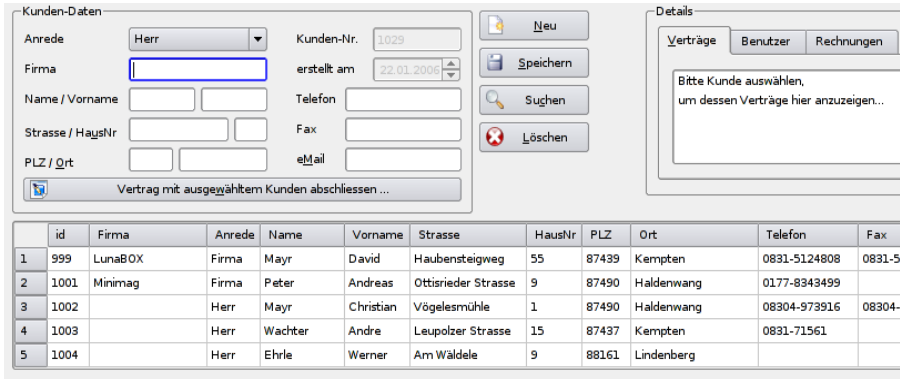
2 Aufbau der Benutzeroberfläche

Um es möglichst übersichtlich zu gestalten, wurde ein Tab-Layout gewählt: die Haupbereiche der Anwendung sind auf verschiedene Tabs (Karteikartenreiter) verteilt.

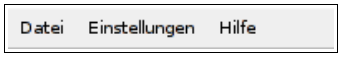


Auf jeder dieser Seiten ist der Aufbau fast gleich:

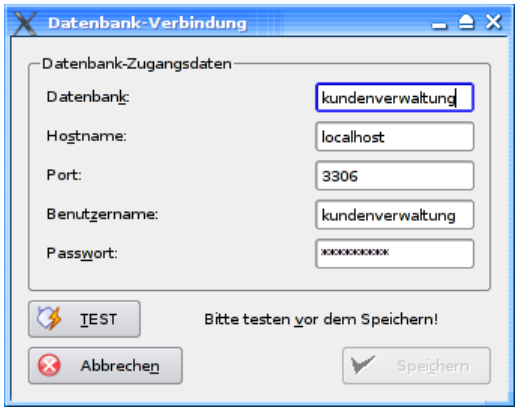
Unten die Datentabelle – in ihr ist auch das direkte Bearbeiten der Daten möglich. Oben links das Eingabeformular und daneben die Aktionsknöpfe Neu, Speichern, Suchen und Löschen. Oben rechts sind auf manchen Seiten Details zum jeweils gewählten Eintrag zu sehen.



Das Hauptmenü ist sehr einfach gehalten:



Unter "Datei" kann man das Programm beenden, unter "Einstellungen" die Konfiguration der Datenbank-Zugangsdaten vornehmen und unter "Hilfe" Informationen zu der Software anzeigen lassen.

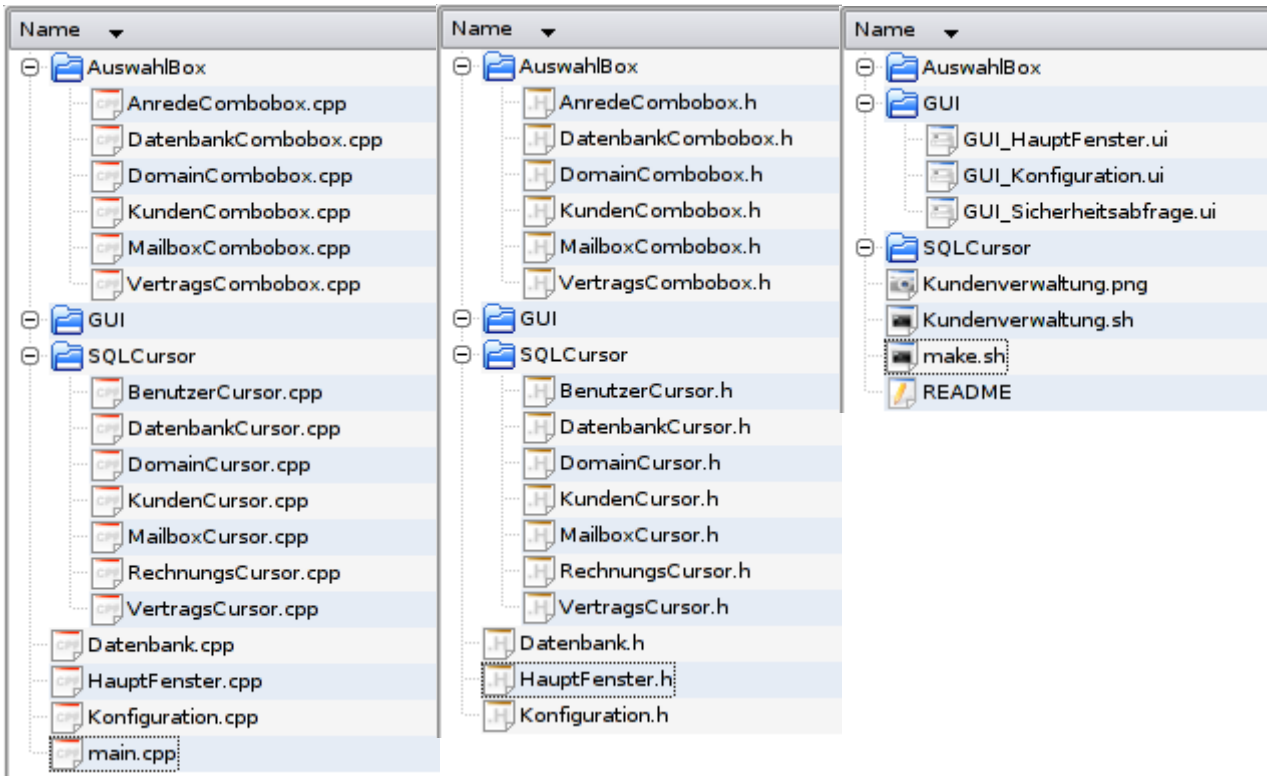


Die Konfiguration muss vor dem Speichern in eine Datei erfolgreich getestet worden sein – danach wird sie bei jedem Start des Programms automatisch geladen und nicht mehr abgefragt.

Rechts ist die Info-Meldung aus dem "Hilfe"-Menü zu sehen.

3 Technische Realisierung

Im folgenden zunächst eine kleine Übersicht über die im Projekt verwendeten Ordner und Dateien, gruppiert nach Dateitypen. Hierbei wurden der übersichtlichkeit halber alle durch QT automatisch erzeugten Dateien entfernt (v.a. im Ordner `GUI`):



3.1 main-Funktion

Die `main()`-Funktion wurde so knapp wie nur möglich gehalten – sie stellt nur einen Splash-Screeen kurz dar und initialisiert dann das HauptFenster. Sie ist die einzige Funktion in der Datei `main.cpp`.

3.2 HauptFenster

Die Klasse Hauptfenster wird von der automatisch durch den QT-Designer & `qmake` erzeugten GUI-Klasse (im Ornder GUI) abgeleitet. Sie beinhaltet die wesentlichen Funktionen der Anwendung.

Allen Methoden wurden ein Prefix entsprechden dem Tab-Namen auf dem sie arbeiten gegeben, so beginnen alle Kunden-Funktionen mit "kunde_...", alle Vertrags-Funktionen mit "vertrag_..." usw.

Für jedes der 7 Tabs gibt es mindestens folgende Funktionen (am Beispiel "Kunde"):

`kunde_`**`init`**`()`, `kunde_`**`neu`**`()`, `kunde_`**`laden`**`()`, `kunde_`**`speichern`**`()`, `kunde_`**`loeschen`**`()`, `kunde_`**`suchen`**`()`

Durch den QT-Designer werden Aktionen (in QT "Signals") am GUI mit Methoden ("Slots") verbunden – z.B. wurde das Signal `clicked()` der Kundentabelle mit der Methode `kunde_laden()` des HauptFensters verbunden.

3.3 Konfiguration

Die Konfiguration ist wie auch das HauptFenster eine im QT-Designer erzeugte Oberfläche die durch die entsprechenden Tools zu Klassen verarbeitet wird, die dann wiederum abgeleitet zum Einsatz kommt.

Das Konfigurations-Fenster wird gleich zu Beginn im Konstruktor des Hauptfensters erzeugt – aber nur angezeigt, wenn die automatisch geladene Konfiguration nicht zu einem erfolgreichen Verbindungtest mit der Datenbank führt. Ansonsten bleibt das Fenster versteckt, bis es über den Menüpunkt "Einstellungen" -> "Datenbank-Konfiguration" aufgerufen wird.

3.4 Datenbank

Die Daten der Datenbank-Verbindung wurden in einer eigenen Klasse gekapselt.

3.5 QsqlCursor – einfacher DB-Zugriff mit QT

Eine sehr vorteilhafte Art der Datenbankverbindung ist in QT mit dem QSqlCursor im Zusammenspiel mit der QDataTable und QsqlForm realisierbar. Die QsqlCursor-Klasse stellt eine Art Abfrage-Zeiger dar, die es mit sehr wenig Aufwand ermöglicht Tabellen einfach zu bearbeiten und an Eingabe-Elemente zu binden.

Zusätzlich ist es mit einem QsqlCursor möglich, eigene generierte Tabellen-Spalten zu erzeugen und dann später in einer QdataTable anzeigen zu lassen.

Alle abgeleiteten QcomboBoxen sind in einem eigenen Verzeichnis mit dem Namen `SQLCursor`.

3.6 QdataTable – Datentabellen

QdataTable ist die Klasse für ein graphisches Element in QT, um Daten tabellarisch darzustellen. Zusätzlich kann an eine QdataTable ein QsqlCursor gebunden werden - dass sie automatisch mit den Werten aus einer Datenbank gefüllt werden.

Ausserdem ist es bei der Kombination aus QdataTable und QsqlCursor möglich, Daten direkt in der Tabelle per Doppelklick zu bearbeiten – jede abgeschlossene Änderung wird sofort automatisch in die Datenbank übertragen.

3.7 QsqlForm – Eingabeformulare

In QT gibt es die Möglichkeit, einen QsqlCursor an ein QsqlForm zu binden – damit werden Eingabefelder mit den Datenbank-Tabellen verknüpft und es wird so eine einfache Datenbearbeitung ermöglicht.

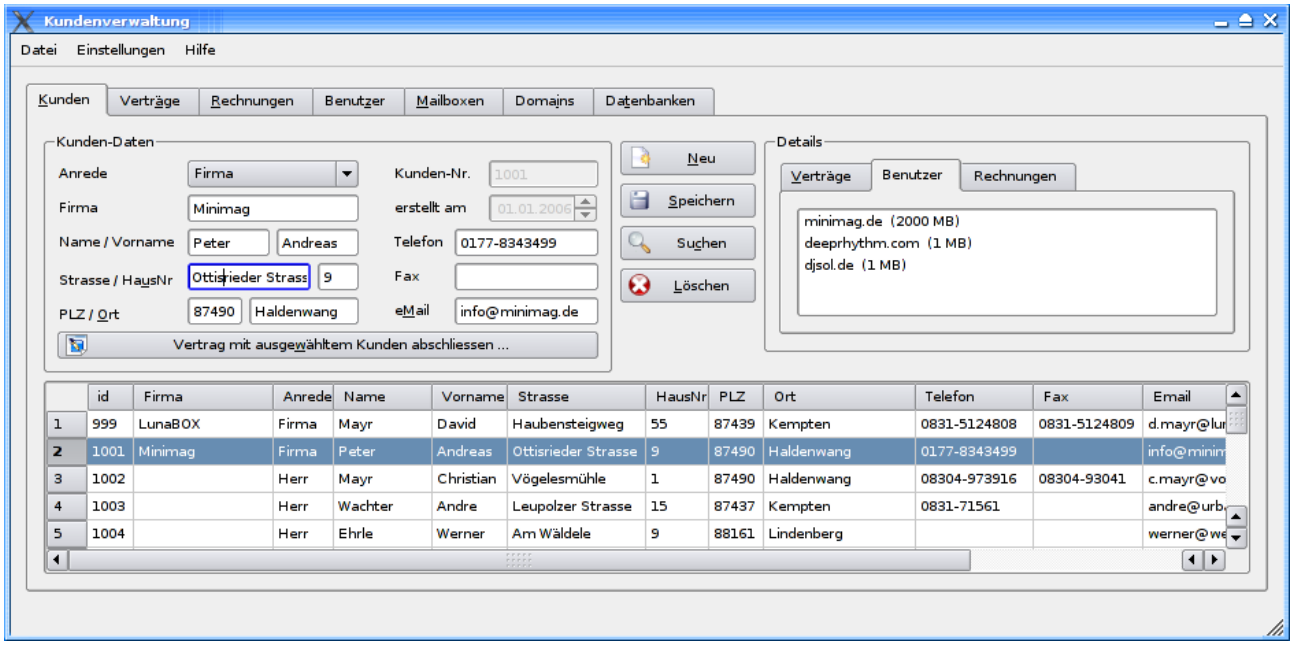
3.8 QcomboBox – spezielle Auswahlboxen

Für die Auswahl z.B. des Kunden, der zu einem Vertrag gehört, oder des Vertrags, zu dem eine Rechnung gehört wurden spezielle Auswahlboxen verwendet, die von der QcomboBox abgeleitet wurden. Sie zeichnen sich daurch aus, dass sie z.B. einen bestimmten Text anzeigen, aber bei einer Abfrage nur die z.B. Vertragsnummer des ausgewählten Vertrags zurückgeben.

Alle abgeleiteten QcomboBoxen sind in einem eigenen Verzeichnis mit dem Namen `Auswahlbox`.

4 Bedienung

4.1 Kunden verwalten



Um einen neuen Kunden anzulegen können zunächst die Eingabefelder links mit dem Knopf "Neu" geleert werden. Dann werden die Daten eingetragen und mit dem Knopf "Speichern" direkt in die Datenbank gespeichert und unten in der Tabelle angezeigt.

Um einen **vorhandenen Kunden zu ändern** gibt es zwei Möglichkeiten: Zum einen kann einfach jedes Tabellenfeld in der Datentabelle unten doppelgeklickt werden und die Daten können dann direkt dort verändert werden. Zum anderen kann ein Kunde in der Tabelle einfach angeklickt werden – dann werden seine Daten automatisch in die Eingabefelder geladen, und nach dem verändern durch den "Speichern"-Knopf wieder abgespeichert werden.

Um einen **Kunden** zu **suchen**, müssen einfach beliebig kombinierte Felder links mit Suchbegriffen gefüllt werden - und der Suchen-Knopf bringt entsprechde Ergebnisse.

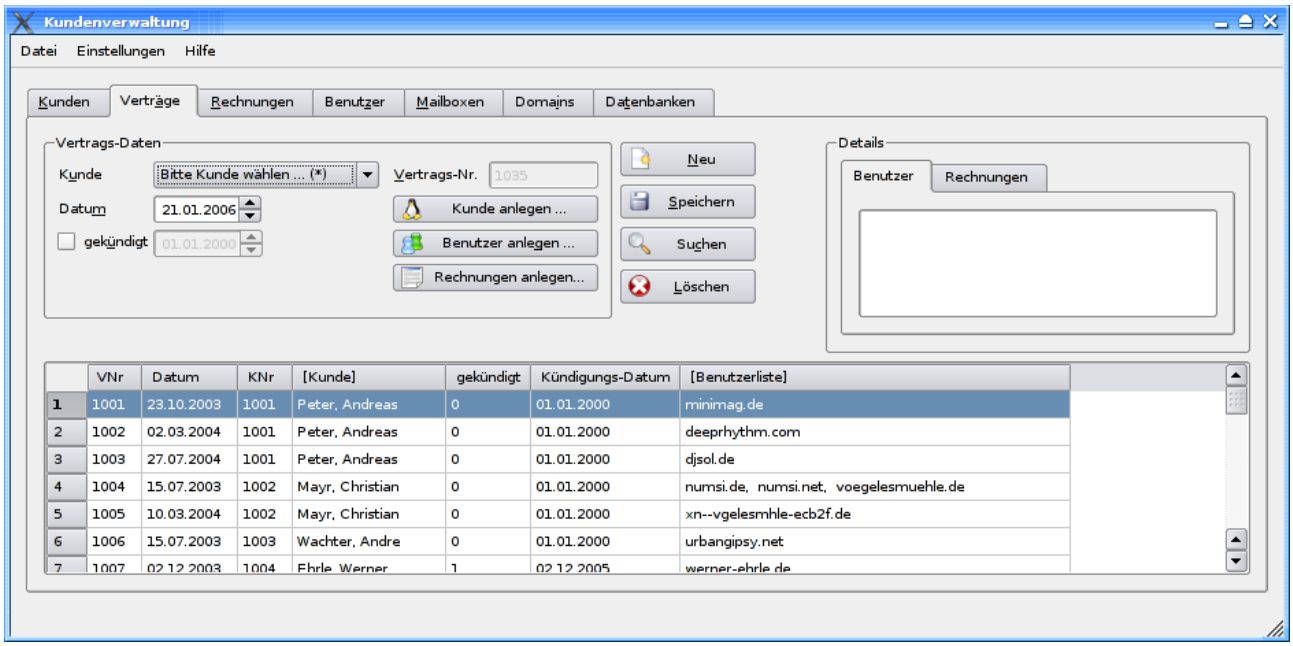
Der **Löschen**-Knopf löscht den zuvor ausgewählte Kunden. Zuvor muss eine Sicheitsabfrage positiv beantwortet werden.

Im rechten oberen Bereich werden bei jedem Laden eines Kunden dessen **Detail-Daten** angezeigt:

- Welche Verträge hat dieser Kunde?
- Welche Benutzer hat dieser Kunde mit seinen Verträgen?
- Welche Rechnungen hatte dieser Kunde bisher?

Da nach dem Anlegen eines Kunden meist das Abschliessen eines Vertrages folgt, wurde zur Vereinfachung der zusätzliche Knopf "Vertrag mit ausgewähltem Kunden abschliessen..." angelegt. Durch ihn wird auf dem Vertrags-Tab der "Neu"-Knopf gedrückt, automatisch der zuletzt gewählte Kunde als Vertragspartner vorausgewählt und auf das Vertrags-Tab gewechselt.

4.2 Verträge verwalten



Um einen neuen Vertrag anzulegen können zunächst die Eingabefelder links mit dem Knopf "Neu" geleert werden. Dann werden die Daten eingetragen und mit dem Knopf "Speichern" direkt in die Datenbank gespeichert und unten in der Tabelle angezeigt.

Um einen **vorhandenen Vertrag zu ändern** gibt es zwei Möglichkeiten: Zum einen kann einfach jedes Tabellenfeld in der Datentabelle unten doppelgeklickt werden und die Daten können dann direkt dort verändert werden. Zum anderen kann ein Vertrag in der Tabelle einfach angeklickt werden – dann werden seine Daten automatisch in die Eingabefelder geladen, und nach dem verändern durch den "Speichern"-Knopf wieder abgespeichert werden.

Um einen Vertrag zu **suchen**, muss einfach ein Kunde links ausgewählt werden - und der Suchen-Knopf bringt alle Verträge des entsprechenden Kunden.

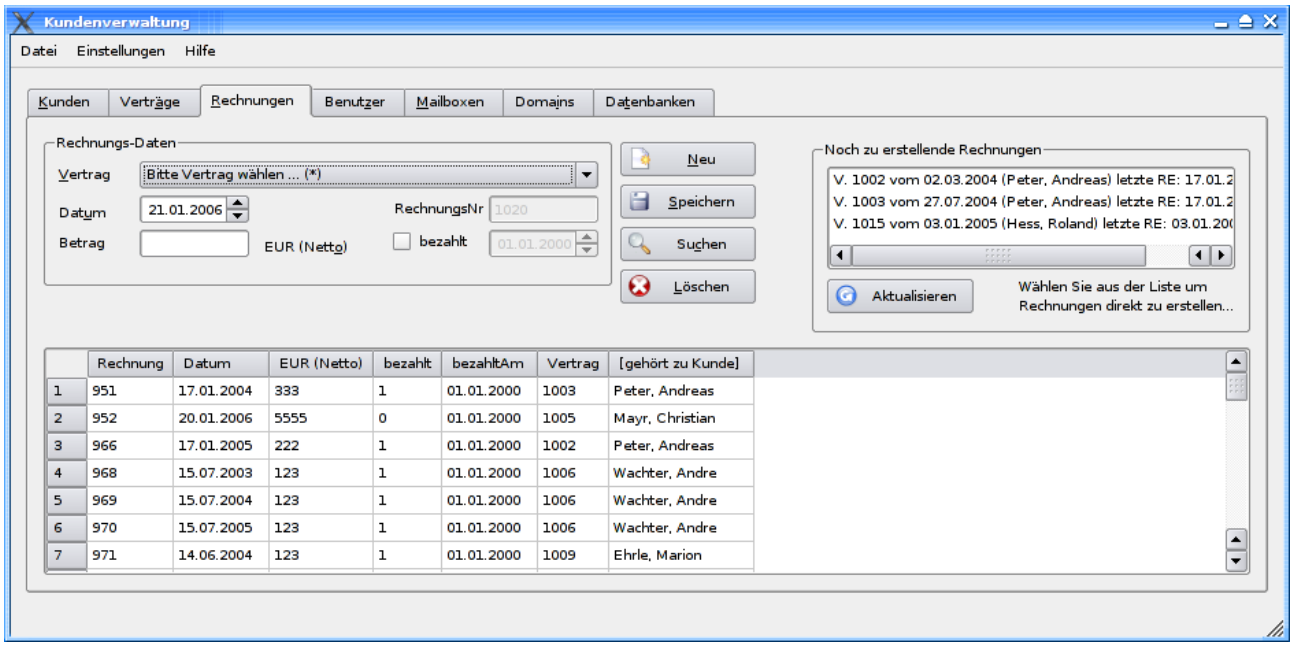
Der **Löschen**-Knopf löscht den zuvor ausgewählte Vertrag. Zuvor muss eine Sicheitsabfrage positiv beantwortet werden.

Im rechten oberen Bereich werden bei jedem Laden eines Vertrags dessen **Detail-Daten** angezeigt:

- Welche Benutzer gehören zu diesem Vertrag?
- Welche Rechnungen gibt es zu diesem Vertrag bisher?

Da nach dem Anlegen eines Vertrags meist das Anlegen von Benutzern oder Rechnungen folgt, die zu diesem Vertrag gehören, wurden zur Vereinfachung die zusätzlichen Knöpfe "Benutzer anlegen..." und "Rechnugen anlegen..." eingebaut. Durch sie wird auf das entsprechende Tab gesprungen, der "Neu"-Knopf gedrückt, automatisch der zuletzt gewählte Vertrag als vorausgewählt und auf das entsprechende Tab gewechselt.

4.3 Rechnungen verwalten



Um eine neue Rechnung anzulegen können zunächst die Eingabefelder links mit dem Knopf "Neu" geleert werden. Dann werden die Daten eingetragen und mit dem Knopf "Speichern" direkt in die Datenbank gespeichert und unten in der Tabelle angezeigt.

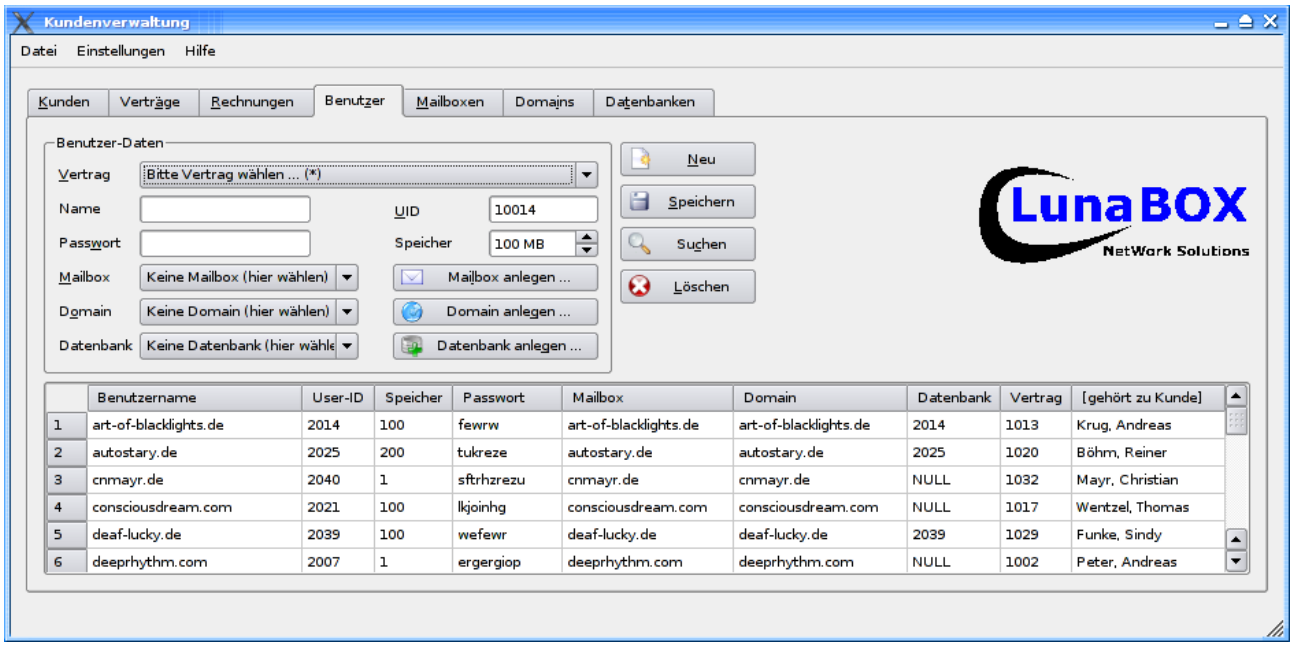
Um eine **vorhandene Rechnung zu ändern** gibt es zwei Möglichkeiten: Zum einen kann einfach jedes Tabellenfeld in der Datentabelle unten doppelgeklickt werden und die Daten können dann direkt dort verändert werden. Zum anderen kann ein Eintrag in der Tabelle einfach angeklickt werden – dann werden seine Daten automatisch in die Eingabefelder geladen, und nach dem verändern durch den "Speichern"-Knopf wieder abgespeichert werden.

Um eine Rechnung zu **suchen**, können folgende Kriterien verwendet werden: Vertrag & Bezahlt-Status. Wenn kein Vertrag gewählt wurde, so wird nur nach dem Bezahlt-Status gesucht – somit lassen sich sehr schnell alle noch offenen Rechnungen ermitteln.

Der **Löschen**-Knopf löscht die zuvor ausgewählte Rechung. Zuvor muss eine Sicheitsabfrage positiv beantwortet werden.

Im rechten oberen Bereich werden alle diejenigen Verträge angezeigt, zu denen es seit über einem Jahr keine Rechnung gibt. Damit wird die **Erinnerung an die Rechnungserstellung** realisiert. Ein Klick auf einen Eintrag erstellt automatisch eine neue Rechung mit dem Datum der letzten Rechung zu diesem Vertrag plus ein Jahr – so muss in den meisten Fällen nur noch der Betrag eingegeben und abgespeichert werden.

4.4 Benutzer verwalten



Um einen neuen Benutzer anzulegen können zunächst die Eingabefelder links mit dem Knopf "Neu" geleert werden. Dann werden die Daten eingetragen und mit dem Knopf "Speichern" direkt in die Datenbank gespeichert und unten in der Tabelle angezeigt.

Um einen **vorhandenen Benutzer zu ändern** gibt es zwei Möglichkeiten: Zum einen kann einfach jedes Tabellenfeld in der Datentabelle unten doppelgeklickt werden und die Daten können dann direkt dort verändert werden. Zum anderen kann ein Eintrag in der Tabelle einfach angeklickt werden – dann werden seine Daten automatisch in die Eingabefelder geladen, und nach dem verändern durch den "Speichern"-Knopf wieder abgespeichert werden.

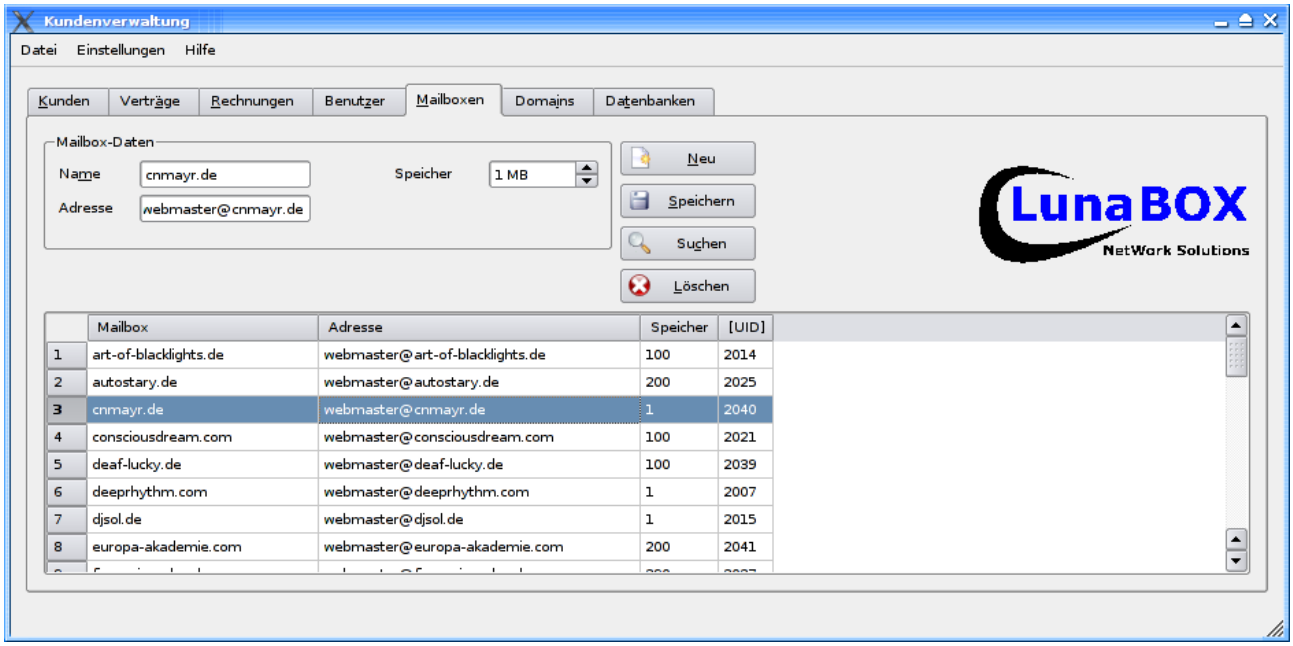
Um einen Benutzer zu **suchen**, kann der Suchbegriff in das Eingabefeld des Benutzernamens eingegeben werden. Der Suchen-Knopf findet dann alle Benutzer die den Suchbegriff in ihrem Benutzernamen haben.

Der **Löschen**-Knopf löscht den zuvor ausgewählte Benutzer. Zuvor muss eine Sicheitsabfrage positiv beantwortet werden.

Da zu jedem Benutzer genau eine Mailbox, evtl. eine Domain und optional auch eine Datenbank gehört, wurde das Anlegen dieser Einträge in den jeweiligen Tabellen stark vereinfacht.

Jeder der "[...] anlegen..."-Knöpfe legt in der entsprechden Tabelle einen neuen Eintrag mit Standardwerten abhängig vom Benutzernamen an und wechselt aus das Tab. So müssen nur noch evtl. Werte ergänzt oder einfach der Speichern-Knopf gedrückt werden – daraufhin wird das Tab wieder zu "Benutzer" gewechselt und der neue Eintrag in der Auswahlbox vorselektiert. So ist das Anlegen einer Domain zu einem Benutzer nicht mehr als genau zwei Klicks...

4.5 Mailboxen verwalten



Um eine neue Mailbox anzulegen können zunächst die Eingabefelder links mit dem Knopf "Neu" geleert werden. Dann werden die Daten eingetragen und mit dem Knopf "Speichern" direkt in die Datenbank gespeichert und unten in der Tabelle angezeigt.

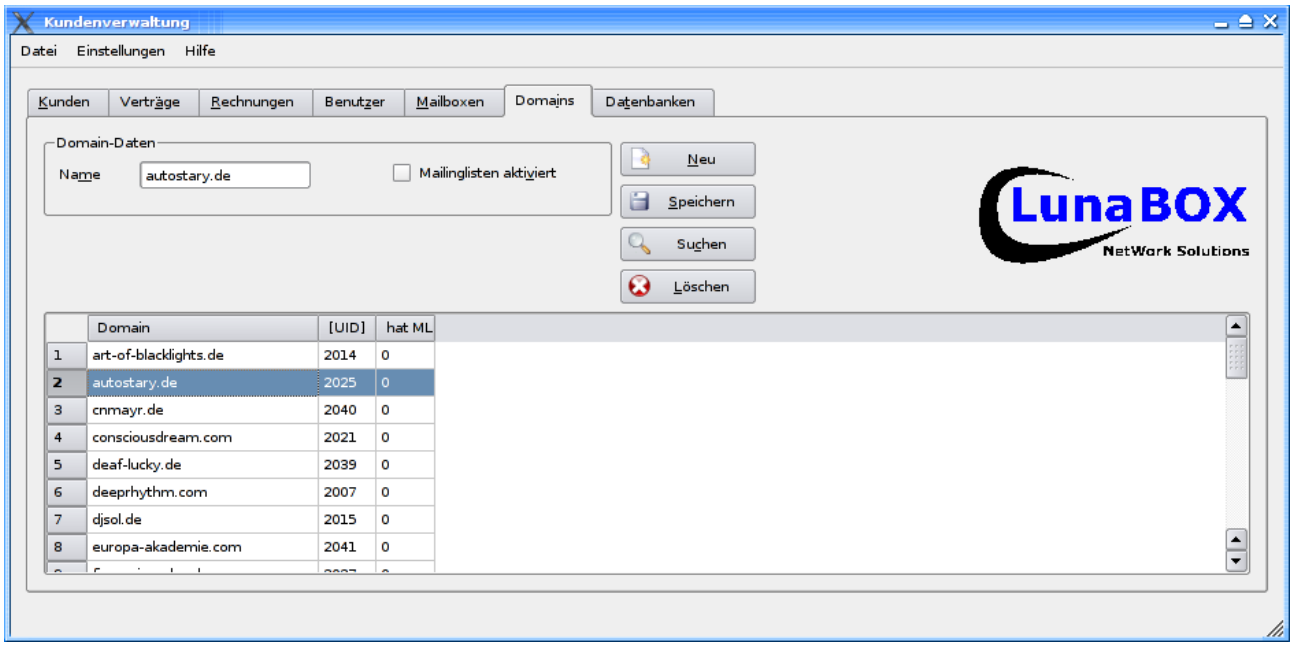
Um eine **vorhandene Mailbox zu ändern** gibt es zwei Möglichkeiten: Zum einen kann einfach jedes Tabellenfeld in der Datentabelle unten doppelgeklickt werden und die Daten können dann direkt dort verändert werden. Zum anderen kann ein Eintrag in der Tabelle einfach angeklickt werden – dann werden seine Daten automatisch in die Eingabefelder geladen, und nach dem verändern durch den "Speichern"-Knopf wieder abgespeichert werden.

Um eine Mailbox zu **suchen**, kann ein Name (oder ein Teil davon) eingegeben werden, und der Suchen-Knopf liefert entsprechende Ergebnisse.

Der **Löschen**-Knopf löscht die zuvor ausgewählte Mailbox. Zuvor muss eine Sicheitsabfrage positiv beantwortet werden.

Der Standardwert für eine Mailbox entspricht in der Regel dem Benutzernamen des Eigentümers.

4.6 Domains verwalten



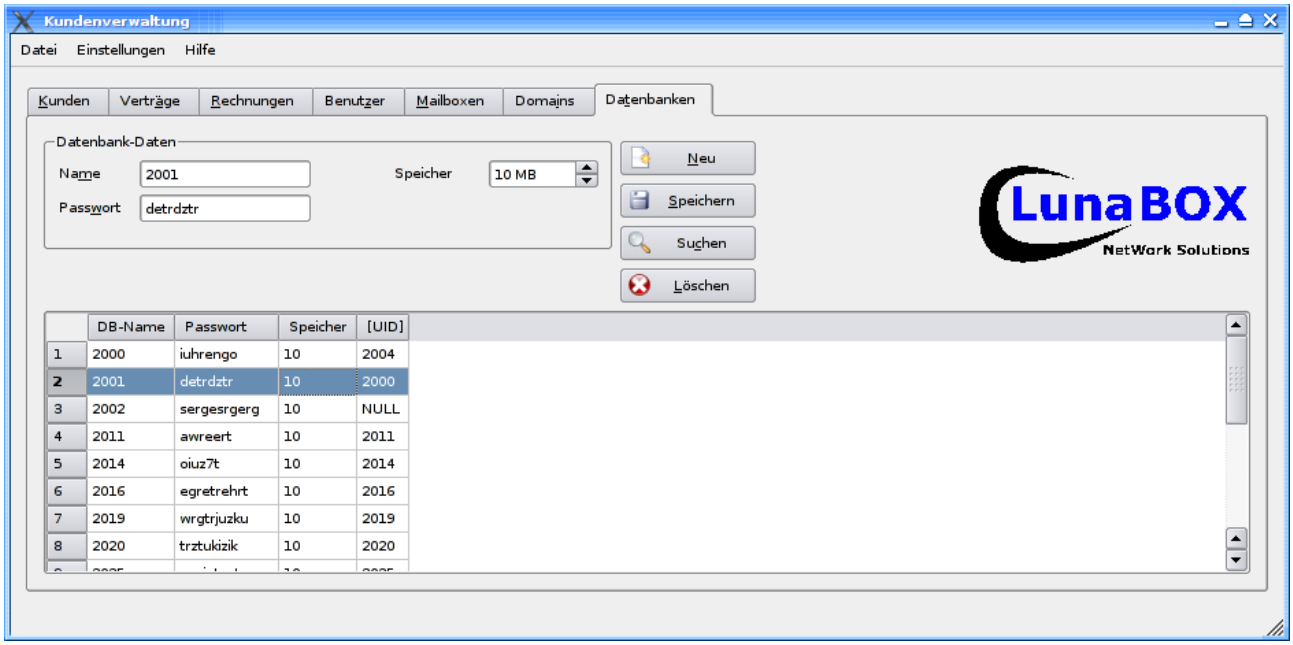
Um eine neue Domain anzulegen können zunächst die Eingabefelder links mit dem Knopf "Neu" geleert werden. Dann werden die Daten eingetragen und mit dem Knopf "Speichern" direkt in die Datenbank gespeichert und unten in der Tabelle angezeigt.

Um eine **vorhandene Domain zu ändern** gibt es zwei Möglichkeiten: Zum einen kann einfach jedes Tabellenfeld in der Datentabelle unten doppelgeklickt werden und die Daten können dann direkt dort verändert werden. Zum anderen kann ein Eintrag in der Tabelle einfach angeklickt werden – dann werden seine Daten automatisch in die Eingabefelder geladen, und nach dem verändern durch den "Speichern"-Knopf wieder abgespeichert werden.

Um eine Domain zu **suchen**, kann ein Name (oder ein Teil davon) eingegeben werden, und der Suchen-Knopf liefert entsprechende Ergebnisse.

Der **Löschen**-Knopf löscht die zuvor ausgewählte Domain. Zuvor muss eine Sicheitsabfrage positiv beantwortet werden.

4.7 Datenbanken verwalten



Um eine neue Datenbank anzulegen können zunächst die Eingabefelder links mit dem Knopf "Neu" geleert werden. Dann werden die Daten eingetragen und mit dem Knopf "Speichern" direkt in die Datenbank gespeichert und unten in der Tabelle angezeigt.

Um eine **vorhandene Datenbank zu ändern** gibt es zwei Möglichkeiten: Zum einen kann einfach jedes Tabellenfeld in der Datentabelle unten doppelgeklickt werden und die Daten können dann direkt dort verändert werden. Zum anderen kann ein Eintrag in der Tabelle einfach angeklickt werden – dann werden seine Daten automatisch in die Eingabefelder geladen, und nach dem verändern durch den "Speichern"-Knopf wieder abgespeichert werden.

Um eine Datenbank zu **suchen**, kann ein Name (oder ein Teil davon) eingegeben werden, und der Suchen-Knopf liefert entsprechende Ergebnisse.

Der **Löschen**-Knopf löscht die zuvor ausgewählte Datenbank. Zuvor muss eine Sicheitsabfrage positiv beantwortet werden.