kivitendo 4.1.0: Installation, Konfiguration, Entwicklung

kivitendo 4.1.0: Installation, Konfiguration, Entwicklung

Inhaltsverzeichnis

1. Aktuelle Hinweise ... 1
2. Installation und Grundkonfiguration ... 2
    2.1. Übersicht ... 2
    2.2. Benötigte Software und Pakete ... 2
        2.2.1. Mindest-Voraussetzung ... 2
        2.2.2. Betriebssystem ... 2
        2.2.3. Benötigte Perl-Pakete installieren ... 3
        2.2.4. Andere Pakete installieren ... 8
    2.3. Installation mittels Ansible auf Ubuntu 22.04 ... 9
    2.4. Manuelle Installation des Programmpaketes ... 9
        2.4.1. Installation mit git ... 9
        2.4.2. Installation über die Github Website ... 10
        2.4.3. Restliche Verzeichnisse ändern und bearbeiten ... 11
    2.5. kivitendo-Konfigurationsdatei ... 11
        2.5.1. Einführung ... 11
        2.5.2. Abschnitte und Parameter ... 11
        2.5.3. Versionen vor 2.6.3 ... 13
    2.6. Anpassung der PostgreSQL-Konfiguration ... 13
        2.6.1. Änderungen an Konfigurationsdateien ... 13
        2.6.2. Erweiterung für servergespeicherte Prozeduren ... 13
        2.6.3. Erweiterung für Trigram Prozeduren ... 14
        2.6.4. Datenbankbenutzer anlegen ... 14
    2.7. Webserver-Konfiguration ... 15
        2.7.1. Konfiguration für FastCGI/FCGI ... 15
        2.7.2. Aktivierung von mod_rewrite/directory_match für git basierte Installationen ... 16
        2.7.3. Weitergehende Konfiguration ... 17
        2.7.4. Aktivierung von Apache2 modsecurity ... 17
    2.8. Der Task-Server ... 17
        2.8.1. Verfügbare und notwendige Konfigurationsoptionen ... 17
        2.8.2. Konfiguration der Mandanten für den Task-Server ... 18
        2.8.3. Automatisches Starten des Task-Servers beim Booten ... 18
        2.8.4. Wie der Task-Server gestartet und beendet wird ... 19
    2.9. Konfiguration der Hintergrund-Jobs ... 19
        2.9.1. SetNumberRange ... 20
        2.9.2. ImportRecordEmails ... 21
        2.9.3. InventoryClearAll ... 23
        2.9.4. CloseQuotations ... 24
    2.10. Benutzerauthentifizierung und Administratorpasswort ... 24
        2.10.1. Grundlagen zur Benutzerauthentifizierung ... 24
        2.10.2. Administratorpasswort ... 24
        2.10.3. Authentifizierungsdatenbank ... 25
        2.10.4. Passwortüberprüfung ... 25
        2.10.5. Name des Session-Cookies ... 27
        2.10.6. Anlegen der Authentifizierungsdatenbank ... 27
    2.11. Mandanten-, Benutzer- und Gruppenverwaltung ... 27
        2.11.1. Zusammenhänge ... 28
        2.11.2. Mandanten, Benutzer und Gruppen ... 28
        2.11.3. Datenbanken anlegen ... 28
        2.11.4. Gruppen anlegen ... 28
        2.11.5. Benutzer anlegen ... 29
        2.11.6. Mandanten anlegen ... 29
    2.12. Drucker- und Systemverwaltung ... 29
        2.12.1. Druckeradministration ... 29
        2.12.2. System sperren / entsperren ... 29
    2.13. E-Mail ... 30

2.13.1. E-Mail-Versand aus kivitendo heraus ... 30
2.13.2. Versendete E-Mails über IMAP exportieren ... 31
2.13.3. E-Mails in kivitendo importieren ... 31
2.14. Drucken mit kivitendo ... 32
2.14.1. Vorlagenverzeichnis anlegen ... 33
2.14.2. Der Druckvorlagensatz marei ... 33
2.14.3. Der Druckvorlagensatz mersiha ... 37
2.14.4. Der Druckvorlagensatz RB ... 40
2.14.5. Der Druckvorlagensatz rev-odt ... 40
2.14.6. Allgemeine Hinweise zu LaTeX Vorlagen ... 40
2.15. OpenDocument-Vorlagen ... 41
2.15.1. Grundeinstellung ... 41
2.15.2. Direkte Erzeugung von PDF-Dateien ... 41
2.15.3. Vorbereitungen ... 42
2.15.4. Schweizer QR-Rechnung mit OpenDocument Vorlagen ... 43
2.16. Nomenklatur ... 44
2.16.1. Datum bei Buchungen ... 44
2.17. Konfiguration zur Einnahmenüberschussrechnung/Bilanzierung: EUR ... 45
2.17.1. Einführung ... 45
2.17.2. Konfigurationsparameter ... 45
2.17.3. Festlegen der Parameter ... 45
2.17.4. Bemerkungen zur Bestandsmethode ... 46
2.17.5. Bekannte Probleme ... 46
2.18. SKR04 19% Umstellung für innergemeinschaftlichen Erwerb ... 46
2.18.1. Einführung ... 46
2.18.2. Konto 3804 manuell anlegen ... 46
2.19. Verhalten des Bilanzberichts ... 50
2.20. Jahresabschluss ... 50
2.21. Erfolgsrechnung ... 51
2.22. Rundung in Verkaufsbelegen ... 51
2.23. Einstellungen pro Mandant ... 52
2.24. kivitendo ERP verwenden ... 52
3. Features und Funktionen ... 53
3.1. Wiederkehrende Rechnungen ... 53
3.1.1. Einführung ... 53
3.1.2. Konfiguration ... 53
3.1.3. Spezielle Variablen ... 54
3.1.4. Auflisten ... 55
3.1.5. Erzeugung der eigentlichen Rechnungen ... 56
3.1.6. Erste Rechnung für aktuellen Monat erstellen ... 56
3.2. Zahlungsverkehr über Bankkonten ... 56
3.2.1. SEPA-Überweisungen ... 56
3.2.2. Kontoauszug Verbuchen mit Buchungsvorlagen und direktem Verbuchen ... 56
3.3. Dokumentenvorlagen und verfügbare Variablen ... 57
3.3.1. Einführung ... 57
3.3.2. Variablen ausgeben ... 57
3.3.3. Verwendung in Druckbefehlen ... 57
3.3.4. Anfang und Ende der Tags verändern ... 58
3.3.5. Zuordnung von den Dateinamen zu den Funktionen ... 58
3.3.6. Sprache, Drucker und E-Mail ... 59
3.3.7. Allgemeine Variablen, die in allen Vorlagen vorhanden sind ... 59
3.3.8. Variablen in Rechnungen ... 65
3.3.9. Variablen in Mahnungen und Rechnungen über Mahngebühren ... 69
3.3.10. Variablen in anderen Vorlagen ... 71
3.3.11. Blöcke, bedingte Anweisungen und Schleifen ... 73
3.3.12. Markup-Code zur Textformatierung innerhalb von Formularen ... 75
3.3.13. Hinweise zur Anrede ... 75
3.4. Excel-Vorlagen ... 75

3.4.1. Zusammenfassung ... 75
3.4.2. Bedienung ... 76
3.4.3. Variablensyntax ... 76
3.4.4. Einschränkungen ... 76
3.5. Mandantenkonfiguration Lager ... 76
3.6. Schweizer Kontenpläne ... 77
3.7. Artikelklassifizierung ... 77
3.7.1. Übersicht ... 77
3.7.2. Basisklassifizierung ... 77
3.7.3. Attribute ... 78
3.7.4. Zwei-Zeichen Abkürzung ... 78
3.8. Dateiverwaltung (Mini-DMS) ... 78
3.8.1. Übersicht ... 78
3.8.2. Struktur ... 79
3.8.3. Anwendung ... 80
3.8.4. Konfigurierung ... 82
3.9. Webshop-Api ... 84
3.9.1. Rechte für die Webshopapi ... 84
3.9.2. Konfiguration ... 84
3.9.3. Webshopartikel ... 85
3.9.4. Bestellimport ... 86
3.9.5. Mapping der Daten ... 88
3.10. ZUGFeRD Rechnungen ... 88
3.10.1. Vorbedingung ... 88
3.10.2. Übersicht ... 88
3.10.3. Erstellen von ZUGFeRD Rechnungen in Kivitendo ... 89
3.10.4. Einlesen von ZUGFeRD Rechnungen in Kivitendo ... 89
3.11. Reklamationen ... 89
3.11.1. Konfiguration des Reklamationsmodul ... 89
3.11.2. Reklamation erfassen ... 90
3.11.3. Reklamationen auswerten ... 91
3.12. Dispositionsmanager/Einkaufshelfer ... 91
3.12.1. So kommen die Artikel in den Einkaufswarenkorb: ... 92
3.12.2. Der Einkaufswarenkorb ... 93
3.13. Inventur-Zählung anlegen und abgleichen ... 94
3.14. Zeiterfassung ... 94
3.14.1. Konfiguration ... 94
3.14.2. Erfassen ... 95
3.14.3. Bericht ... 95
3.14.4. Konvertierung zu Lieferscheinen ... 95
3.15. Ticketsystem-Schnittstelle ... 97
3.15.1. Atlassian Jira ... 97
4. Entwicklerdokumentation ... 99
4.1. Globale Variablen ... 99
4.1.1. Wie sehen globale Variablen in Perl aus? ... 99
4.1.2. Warum sind globale Variablen ein Problem? ... 99
4.1.3. Kanonische globale Variablen ... 100
4.1.4. Ehemalige globale Variablen ... 103
4.2. Entwicklung unter FastCGI ... 103
4.2.1. Allgemeines ... 103
4.2.2. Programmende und Ausnahmen ... 104
4.2.3. Globale Variablen ... 104
4.2.4. Performance und Statistiken ... 104
4.3. Entwicklung unter CGI ... 104
4.4. Programmatische API-Aufrufe ... 105
4.4.1. Einführung ... 105
4.4.2. HTTP-»Basic«-Authentifizierung ... 105
4.4.3. Beispiele ... 105

4.5. SQL-Upgradedateien 105
4.5.1. Einführung 105
4.5.2. Format der Kontrollinformationen 105
4.5.3. Format von in Perl geschriebenen Datenbankupgradescripten 106
4.5.4. Hilfsscript dbupgrade2_tool.pl 107
4.6. Translations and languages 108
4.6.1. Introduction 108
4.6.2. Character set 108
4.6.3. File structure 108
4.7. Die kivitendo-Test-Suite 110
4.7.1. Einführung 110
4.7.2. Voraussetzungen 110
4.7.3. Existierende Tests ausführen 111
4.7.4. Bedeutung der verschiedenen Test-Scripte 111
4.7.5. Neue Test-Scripte erstellen 111
4.8. Stil-Richtlinien 112
4.9. Design 4.0 Less/CSS Anpassen 115
4.9.1. Einführung 115
4.9.2. Less kompilieren 115
4.9.3. Lokale Anpassungen (nicht Entwicklung) 115
4.9.4. Weiteres 117
4.10. Dokumentation erstellen 117
4.10.1. Einführung 117
4.10.2. Benötigte Software 117
4.10.3. PDFs und HTML-Seiten erstellen 118
4.10.4. Einchecken in das Git-Repository 118

Abbildungsverzeichnis

4.1. Screenshot mit einigen Variablennamen beschriftet 116

Tabellenverzeichnis

2.1. Dateinamen der Belegtypen 37
2.2. Druckvarianten der Belegtypen (sofern Varianten vorhanden sind) 38

1

Aktuelle Hinweise

Aktuelle Installations- und Konfigurationshinweise gibt es:

- im Community-Forum: https://forum.kivitendo.de
- im Kunden-Forum: https://www.kivitendo.de/redmine/projects/forum/boards/
- in der doc/UPGRADE Datei im doc-Verzeichnis der Installation
- Im Schulungs- und Dienstleistungsangebot der entsprechenden kivitendo-Partner: https://www.kivitendo.de/partner.html

2

Installation und Grundkonfiguration

2.1. Übersicht

Die Installation von kivitendo umfasst mehrere Schritte. Die folgende Liste kann sowohl für Neulinge als auch für alte Hasen als Übersicht und Stichpunktliste zum Abhaken dienen, um eine Version mit minimalen Features möglichst schnell zum Laufen zu kriegen.

1. *Voraussetzungen überprüfen*: kivitendo benötigt gewisse Ressourcen und benutzt weitere Programme. Das Kapitel "Abschnitt 2.2, „Benötigte Software und Pakete“ [2]" erläutert diese. Auch die Liste der benötigten Perl-Module befindet sich hier.

2. *Installation von kivitendo*: Diese umfasst die "Manuelle Installation des Programmpaketes [9]" sowie grundlegende Einstellungen, die der "Abschnitt 2.5, „kivitendo-Konfigurationsdatei“ [11]" erläutert.

3. *Konfiguration externer Programme*: hierzu gehören die Datenbank ("Abschnitt 2.6, „Anpassung der PostgreSQL-Konfiguration“ [13]") und der Webserver ("Abschnitt 2.7, „Webserver-Konfiguration“ [15]").

4. *Benutzerinformationen speichern können*: man benötigt mindestens eine Datenbank, in der Informationen zur Authentifizierung sowie die Nutzdaten gespeichert werden. Wie man das als Administrator macht, verrät "Abschnitt 2.10, „Benutzerauthentifizierung und Administratorpasswort“ [24]".

5. *Benutzer, Gruppen und Datenbanken anlegen*: wie dies alles zusammenspielt erläutert "Abschnitt 2.11, „Mandanten-, Benutzer- und Gruppenverwaltung“ [27]".

6. *Los geht's*: alles soweit erledigt? Dann kann es losgehen: "Abschnitt 2.24, „kivitendo ERP verwenden“ [52]"

Alle weiteren Unterkapitel in diesem Kapitel sind ebenfalls wichtig und sollten vor einer ernsthaften Inbetriebnahme gelesen werden.

2.2. Benötigte Software und Pakete

2.2.1. Mindest-Voraussetzung

kivitendo benötigt einen Webserver der Perl unterstüzt und PostgreSQL als Datenbank in einer entsprechenden Minimalversion.

- PostgreSQL Version 14
- Perl Version 5.34

2.2.2. Betriebssystem

kivitendo ist für Linux konzipiert, und sollte auf jedem unixoiden Betriebssystem zum Laufen zu kriegen sein. Getestet ist diese Version im speziellen auf Debian und Ubuntu, grundsätzlich wurde bei der Auswahl der Pakete aber darauf Rücksicht genommen, dass es ohne große Probleme auf den derzeit aktuellen verbreiteten Distributionen läuft.

Mitte 2025 (ab Version 3.9.2) empfehlen wir:

- Debian
  - 12.0 "Bookworm"
  - 13.0 "Trixie"
- Ubuntu
  - 22.04 "Jammy Jellyfish" LTS
  - 24.04 "Noble Numbat" LTS
- openSUSE Leap 15.6 und SUSE Linux Enterprise Server 15 SP6
- Fedora 39 - 41

2.2.3. Benötigte Perl-Pakete installieren

Zum Betrieb von kivitendo werden zwingend ein Webserver (meist Apache) und ein Datenbankserver (PostgreSQL) in einer aktuellen Version (s.a. Liste der unterstützten Betriebssysteme) benötigt.

Zusätzlich benötigt kivitendo einige Perl-Pakete, die nicht Bestandteil einer Standard-Perl-Installation sind. Um nach der Installation von Kivitendo zu überprüfen, ob die erforderlichen Pakete installiert und aktuell genug sind, wird ein Script mitgeliefert, das wie folgt aufgerufen wird:

```
./scripts/installation_check.pl
```

Anmerkung

Das Paket `List::MoreUtils` wird benötigt um das Script ausführen zu können!

Die vollständige Liste der benötigten Perl-Module lautet:

- `Algorithm::CheckDigits`
- `Archive::Zip`
- `CGI`
- `Clone`
- `Config::Std`
- `CryptX`
- `Daemon::Generic`
- `DateTime`
- `DateTime::Event::Cron`
- `DateTime::Format::Strptime`
- `DateTime::Set`
- `DBI`
- `DBD::Pg`

- `Digest::SHA`
- `Email::Address`
- `Email::MIME`
- `Encode::IMAPUTF7`
- `Exception::Class`
- `FCGI`
- `File::Copy::Recursive`
- `File::Flock`
- `File::MimeInfo`
- `File::Slurp`
- `GD`
- `HTML::Parser`
- `HTML::Restrict`
- `Image::Info`
- `Imager`
- `Imager::QRCode`
- `IPC::Run`
- `JSON`
- `List::MoreUtils`
- `List::UtilsBy`
- LWP::Authen::Digest
- LWP::UserAgent
- `Mail::IMAPClient`
- `Net::SMTP::SSL` (optional, bei E-Mail-Versand über SSL; siehe Abschnitt "E-Mail-Versand über einen SMTP-Server [30]")
- `Net::SSLGlue` (optional, bei E-Mail-Versand über TLS; siehe Abschnitt "E-Mail-Versand über einen SMTP-Server [30]")
- `Math::Round`
- `Params::Validate`
- `PBKDF2::Tiny`
- `PDF::API2`
- `Regexp::IPv6`
- `Rest::Client`

- `Rose::Object`
- `Rose::DB`
- `Rose::DB::Object` Version 0.788 oder neuer
- `Set::Infinite`
- `String::ShellQuote`
- `Sort::Naturally`
- `Template`
- `Text::CSV_XS`
- `Text::Iconv`
- `Text::Unidecode`
- `Try::Tiny`
- `URI`
- `XML::Writer`
- `XML::LibXML`
- `YAML::XS` oder `YAML`
- `UUID::Tiny`

In der Version v4.1.0 sind keine neuen Pakete hinzugekommen.

In der Version v4.0.0 ist folgendes Paket hinzugekommen: `CryptX`

In der Version v3.9.2 sind keine neuen Pakete hinzugekommen.

In der Version v3.9.1 sind keine neuen Pakete hinzugekommen.

In der Version v3.9.0 sind die folgenden Pakete hinzugekommen: `Mail::IMAPClient`, `Encode::IMAPUTF7`. Nicht mehr benötigt wird `CAM::PDF`.

In der Version v3.8.0 sind keine neuen Pakete hinzugekommen.

In der Version v3.7.0 sind keine neuen Pakete hinzugekommen.

Seit Version größer v3.6.0 sind die folgenden Pakete hinzugekommen: `IPC::Run`

Seit Version größer v3.5.8 sind die folgenden Pakete hinzugekommen: `Imager`, `Imager::QRCode` `Rest::Client` `Term::ReadLine::Gnu`

Seit Version größer v3.5.6 sind die folgenden Pakete hinzugekommen: `Try::Tiny`, `Math::Round`

Seit Version größer v3.5.6 sind die folgenden Pakete hinzugekommen: `XML::LibXML`

Seit Version größer v3.5.3 sind die folgenden Pakete hinzugekommen: `Exception::Class`

Seit Version größer v3.5.1 sind die folgenden Pakete hinzugekommen: `Set::Infinite`, `List::UtilsBy`, `DateTime::Set`, `DateTime::Event::Cron` `Daemon::Generic`, `DateTime::Event::Cron`, `File::Flock`, `File::Slurp`

Seit Version größer v3.5.0 sind die folgenden Pakete hinzugekommen: `Text::Unidecode`, `LWP::Authen::Digest`, `LWP::UserAgent`

Seit Version v3.4.0 sind die folgenden Pakete hinzugekommen: `Algorithm::CheckDigits`, `PBKDF2::Tiny`

Seit Version v3.2.0 sind die folgenden Pakete hinzugekommen: `GD`, `HTML::Restrict`, `Image::Info`

Seit v3.0.0 sind die folgenden Pakete hinzugekommen: `File::Copy::Recursive`.

Seit v2.7.0 sind die folgenden Pakete hinzugekommen: `Email::MIME`, `Net::SMTP::SSL`, `Net::SSLGlue`.

Gegenüber Version 2.6.0 sind zu dieser Liste 2 Pakete hinzugekommen, `URI` und `XML::Writer` sind notwendig. Ohne startet kivitendo nicht.

Gegenüber Version 2.6.3 ist `JSON` neu hinzugekommen.

`Email::Address` und `List::MoreUtils` wurden aus dem Lieferumfang entfernt. Es wird empfohlen diese Module zusammen mit den anderen als Bibliotheken zu installieren.

Gegenüber Version 2.6.1 sind `parent`, `DateTime`, `Rose::Object`, `Rose::DB` und `Rose::DB::Object` neu hinzugekommen. `IO::Wrap` wurde entfernt.

2.2.3.1. Debian und Ubuntu

Für Debian und Ubuntu stehen die meisten der benötigten Pakete als Debian-Pakete zur Verfügung. Sie können mit folgendem Befehl installiert werden:

```
apt install  apache2 libarchive-zip-perl libclone-perl \
  libconfig-std-perl libdatetime-perl libdbd-pg-perl libdbi-perl \
  libemail-address-perl  libemail-mime-perl libfcgi-perl libjson-perl \
  liblist-moreutils-perl libnet-smtp-ssl-perl libnet-sslglue-perl \
  libparams-validate-perl libpdf-api2-perl librose-db-object-perl \
  librose-db-perl librose-object-perl libsort-naturally-perl \
  libstring-shellquote-perl libtemplate-perl libtext-csv-xs-perl \
  libtext-iconv-perl liburi-perl libxml-writer-perl libyaml-perl \
  libimage-info-perl libgd-gd2-perl libapache2-mod-fcgid \
  libfile-copy-recursive-perl postgresql libalgorithm-checkdigits-perl \
  libcrypt-pbkdf2-perl git libcgi-pm-perl libtext-unidecode-perl libwww-perl \
  postgresql-contrib poppler-utils libhtml-restrict-perl \
  libdatetime-set-perl libset-infinite-perl liblist-utilsby-perl \
  libdaemon-generic-perl libfile-flock-perl libfile-slurp-perl \
  libfile-mimeinfo-perl libpbkdf2-tiny-perl libregexp-ipv6-perl \
  libdatetime-event-cron-perl libexception-class-perl \
  libxml-libxml-perl libtry-tiny-perl libmath-round-perl \
  libimager-perl libimager-qrcode-perl librest-client-perl libipc-run-perl \
  libencode-imaputf7-perl libmail-imapclient-perl libuuid-tiny-perl \
  libcryptx-perl libauthen-sasl-perl
```

Sollten Pakete nicht zu Verfügung stehen, so können diese auch mittels CPAN installiert werden. Ferner muss für Ubuntu das Repository "Universe" aktiv sein (s.a. Anmerkungen).

Anmerkung

Die Perl Pakete für Ubuntu befinden sich im "Universe" Repository. Falls dies nicht aktiv ist, kann dies mit folgendem Aufruf aktiviert werden:

```
add-apt-repository universe
```

2.2.3.2. Fedora

Für Fedora stehen die meisten der benötigten Perl-Pakete als RPM-Pakete zur Verfügung. Sie können mit folgendem Befehl installiert werden:

```
dnf install httpd mod_fcgid postgresql-server postgresql-contrib\
  perl-Algorithm-CheckDigits perl-Archive-Zip perl-CPAN perl-Class-XSAccessor \
  perl-Clone perl-Config-Std perl-DBD-Pg perl-DBI perl-Daemon-Generic \
  perl-DateTime perl-DateTime-Set perl-Email-Address perl-Email-MIME perl-FCGI \
  perl-File-Copy-Recursive perl-File-Flock perl-File-MimeInfo perl-File-Slurp \
  perl-GD perl-HTML-Restrict perl-JSON perl-List-MoreUtils perl-List-UtilsBy \
  perl-Net-SMTP-SSL perl-Net-SSLGlue perl-PBKDF2-Tiny perl-PDF-API2 \
  perl-Params-Validate perl-Regexp-IPv6 perl-Rose-DB perl-Rose-DB-Object \
  perl-Rose-Object perl-Sort-Naturally perl-String-ShellQuote \
  perl-Template-Toolkit perl-Text-CSV_XS perl-Text-Iconv perl-URI perl-XML-Writer \
  perl-YAML perl-libwww-perl perl-CryptX
```

2.2.3.3. openSUSE Leap 15.4 und SUSE Linux Enterprise Server 15

Für openSUSE Leap 15.4, sowie SLES 15 stehen alle benötigten Perl-Pakete als RPM-Pakete zur Verfügung.

Damit diese installiert werden können, muß das System die erforderlichen Repositories kennen und Zugriff über das Internet darauf haben.

Daher machen wir die Repositories dem System bekannt.

Um die zusätzlichen Repositories für die Installation zur Verfügung zu stellen, kann man diese mit YaST oder auch in einem Terminal auf der Konsole bekannt geben. Wir beschränken uns hier mit der Eingabe auf der Konsole. In den allermeisten Fällen verwenden die Administratoren eine sichere SSH-Verbindung zum zu administrierenden Server.

Dazu geben wir folgenden Befehl ein:

Das ERP kivitendo Repository:

```
zypper addrepo -f -p 90 \
  http://download.opensuse.org/repositories/Application:/ERP:/kivitendo/15.4/ \
  "OSS-15.4-App_ERP_kivitendo"
```

Das ERP kivitendo Evaluation Repository:

```
zypper addrepo -f -p 90 \
  http://download.opensuse.org/repositories/Application:/ERP:/kivitendo:/eval/15.4/ \
  "OSS-15.4-App_ERP_kivitendo_eval"
```

Danach geben wir noch die beiden folgenden Befehle ein:

```
zypper clean
```

```
zypper refresh
```

Sollte zypper eine Meldung ausgeben, ob der Repository Key abgelehnt, nicht vertraut oder für immer akzeptiert werden soll, ist die Beantwortung durch drücken der "i" Taste am besten geeignet. Wer noch mehr über zypper erfahren möchte, kann sich einmal die zypper Hilfe anschauen.

```
zypper --help
```

Anmerkung

Offiziell wird von openSUSE nur noch Versionen ab 15.4 unterstützt. Die SuSE Macher haben ab Version 15.x einen großen Umbau in der Verwaltung der Pakete vorgenommen, das heißt, der Paketumfang ist der SLE

15 als Programmunterbau angepasst. Dies gilt besonders der openSUSE Distribution. Es gibt ja einmal die openSUSE Distri und die professionelle SLE (SUSE Linux Enterprise) Version. Dadurch sind viele Pakete aus dem ursprünglich nur für die openSUSE geltenen Repository enfernt worden, aber auch viele auf aktuellem Stand gehalten. Mit der openSUSE Leap 15.4 ist es dem Benutzer ganz einfach gemacht worden zur professionellen SUSE Linux Enterprise zu wechseln, da die RPM Paketbasis identisch ist.

Ob openSUSE Leap oder SLE, man kann die Distribution auch als reine Text Version, also ohne KDE Oberfläche aufsetzen. Vorteil hierbei ist, dass weniger Balast und unnötige Pakte installiert werden. Diese Variante wird gerne auch als 'Server', 'minimale Server' oder auch 'minmal X' Variante bezeichnet.

Als Administrator hatte man jedoch auch immer die Möglichkeit, bei der Installation der Distribution die KDE (graphische) Oberfläche zu aktivieren. In dieser Konstellation hat man die Möglichkeit, eine VNC Verbindung vom administrativen Client zu verwenden. Ist das nicht eingerichtet, arbeitet der Admin dann direkt am Bildschirm des Servers. Nun loggen wir uns am Server direkt ein, starten Yast2 in einer Konsole wie folgt:

yast2 return.

Oder über die Menüführung wie folgt: Ein Klick auf das runde Icon, ganz links unten in der Menüleiste, dann die Maus verfahren auf System und YaST.

Im weiteren Verlauf der Installation, beschränken wir uns mit dem Installations Werkzeug zypper. Zypper ist ein Komandozeilen basiertes Installations Tool, welches bei openSUSE Standard ist. Zypper weist ein etwas eigenartiges Verhalten auf, dass sich in etwa wie folgt darstellt. Hat man die Repositories eingerichtet, kann man diese mit Yast als auch mit Zypper benutzen, setzt man einen Befehl wie etwa: zypper up ab, so findet zypper mehr neuere Programmversionen als Yast. Ich habe im allgemeinen noch keine Nachteile damit erlebt.

Programmpakete können mit folgendem Befehl installiert werden:

zypper install Paketname

Es wird empfohlen zusätzliche Pakete nicht direkt mit CPAN zu installieren, da man diese auch über andere Repositories beziehen kann, die bei openSUSE zur Verfügung stehen. Dadurch hat man den Vorteil, dass die Pakete mit YaST verwaltet werden, also wieder deinstalliert oder durch neuere ersetzt werden können. Zudem kann man auch noch eventuelle Bugs an openSUSE senden und diese dem Maintainer melden.

```
zypper install kivitendo-erp
```

Für Entwickler installiert man noch das folgende Paket:

```
zypper install kivitendo-erp-devel
```

2.2.4. Andere Pakete installieren

- `poppler-utils` 'pdfinfo' zum Erkennen der Seitenanzahl bei der PDF-Generierung
- `Postgres Trigram-Index` Für datenbankoptimierte Suchanfragen. Bspw. im Paket `postgresql-contrib` enthalten

Debian und Ubuntu:

```
apt install postgresql-contrib poppler-utils
```

Fedora:

```
dnf install poppler-utils postgresql-contrib
```

openSUSE:

```
Nicht notwendig, da poppler-utils bereits i.v.m. kivitendo-erp installiert wird
```

2.3. Installation mittels Ansible auf Ubuntu 22.04

Ansible ist ein Open-Source-Automatisierungstool, das verwendet wird, um die Bereitstellung, Konfiguration und Verwaltung von IT-Systemen zu automatisieren. Dabei führt es Aufgaben über SSH auf entfernten Rechnern (Hosts) aus. Die Aufgaben werden dabei deklarativ als YAML Dateien, den sogenannten Playbooks, übergeben. Benötigt wird lediglich python und ansible ab Version 2.10.

Alle benötigten Konfigurationsdateien und das Playbook sind auf dem kivitendo github Account unter dem Repository namens kivitendo-ansible[1] verfügbar. Das Repo kann lokal mit folgendem git-Befehl "geklont" werden:

```
git clone https://github.com/kivitendo/kivitendo-ansible
```

In diesem Repository befindet sich auch eine Readme.md, die aktuelle Installationshinweise auf englisch enthält.

Um die Installation zu starten, wechseln Sie dann zunächst in den erstellten Ordner 'kivitendo-ansible' und editieren die Datei inventory.

```
cd kivitendo-ansible
vim inventory
```

Der/die Hosts auf denen Kivitendo installiert werden soll wird dann in dieser Datei ('inventory') eingefügt, bspw. die

```
192.168.1.121
```

oder als Namen:

```
kivi.meine-lokale-domaene.de
```

Zusätzliche Parameter könnten erforderlich sein, bspw. der Benutzer für den SSH-Login:

```
kivi.meine-lokale-domaene.de ansible_user=myuser
```

Danach kann das Playbook mittels:

```
ansible-playbook --ask-become main.yml
```

ausgeführt werden, je nach Konfiguration wird man noch aufgefordert das BECOME password einzutragen, hierbei handelt es sich um das Passwort des Benutzers, über den die Installation dann auf dem Ziel-Rechner ausgeführt wird.

Nach erfolgreichen Ausführen des Playbooks ist Kivitendo dann über den Browser erreichbar unter `http://<IP des rechners>/kivitendo-erp/`

Nun muss noch eine Datenbank für Benutzer und Mandanten angelegt werden. Dies kann über die Weboberfläche getan werden, indem man sich mit dem Passwort `admin123` in der Administrationsoberfläche anmeldet. Weitere Details siehe Abschnitt "Abschnitt 2.10, „Benutzerauthentifizierung und Administratorpasswort“ [24]"

2.4. Manuelle Installation des Programmpaketes

2.4.1. Installation mit git

Wir empfehlen eine Installation mittels des Versionsmanager git. Hierfür muss ein git-Client installiert sein. Damit ist man sehr viel flexibler für zukünftige Upgrades. Installations-Anleitung (bitte die Pfade anpassen) bspw. wie folgt:

[1] https://github.com/kivitendo/kivitendo-ansible

```
cd /var/www/
git clone https://github.com/kivitendo/kivitendo-erp.git
cd kivitendo-erp/
git checkout `git tag -l | egrep -ve "(alpha|beta|rc)" | tail -1`
```

Erläuterung: Der Befehl wechselt zur letzten Stable-Version (git tag -l listet alle Tags auf, das egrep schmeisst alle Einträge mit alpha, beta oder rc raus und das tail gibt davon den obersten Treffer zurück). Sehr sinnvoll ist es, direkt im Anschluss einen eigenen Branch zu erzeugen, um bspw. seine eigenen Druckvorlagen-Anpassungen damit zu verwalten. Hierfür reicht ein simples

```
  git checkout -b meine_eigenen_änderungen
```

nach dem letzten Kommando (weiterführende Informationen Git Magic[2]).

Ein beispielhafter Workflow für Druckvorlagen-Anpassungen von 3.4.1 nach 3.5:

```
$ git clone https://github.com/kivitendo/kivitendo-erp.git
$ cd kivitendo-erp/
$ git checkout release-3.4.1                # das ist ein alter release aus dem wir starte
$ git checkout -b meine_eigene_änderungen   # unser lokaler branch - unabhängig von allen
$ git add templates/mein_druck              # das sind unsere druckvorlagen inkl. produktb
$ git commit -m "juhu tolle änderungen"

[meine_aenderungen 1d89e41] juhu tolle ändernungen
 4 files changed, 380 insertions(+)
 create mode 100644 templates/mein_druck/img/webdav/tesla.png
 create mode 100644 templates/mein_druck/mahnung.tex
 create mode 100644 templates/mein_druck/zahlungserinnerung_zwei.tex
 create mode 100644 templates/mein_druck/zahlungserinnerung_zwei_invoice.tex

# 5 Jahre später ...
# webserver abschalten!

$ git checkout master
$ git pull                                  # oder git fetch und danach ein stable release
$ git checkout meine_eigenen_änderungen
$ git rebase master

Zunächst wird der Branch zurückgespult, um Ihre Änderungen
darauf neu anzuwenden ...
Wende an: juhu tolle änderungen
$ service apache2 restart                   # webserver starten!
```

2.4.2. Installation über die Github Website

Der aktuelle Stable-Release, bzw. beta Release wird bei github gehostet und kann hier[3] heruntergeladen werden.

Das aktuelleste kivitendo ERP-Archiv (`kivitendo-erp-*.tgz`) wird dann im Dokumentenverzeichnis des Webservers (z.B. `/var/www/html/`, `/srv/www/htdocs` oder `/var/www/`) entpackt:

```
cd /var/www
tar xvzf kivitendo-erp-*.tgz
```

Wechseln Sie in das entpackte Verzeichnis:

```
cd kivitendo-erp
```

[2] http://www-cs-students.stanford.edu/~blynn/gitmagic/index.html
[3] https://github.com/kivitendo/kivitendo-erp/tags

Alternativ können Sie auch einen Alias in der Webserverkonfiguration benutzen, um auf das tatsächliche Installationsverzeichnis zu verweisen.

2.4.3. Restliche Verzeichnisse ändern und bearbeiten

Die folgenden Schritte müssen nach der Installation mittels git oder der Github Website angewendet werden.

Bei einer Neuinstallation von Version 3.1.0 oder später muss das WebDAV Verzeichnis derzeit manuell angelegt werden:

```
mkdir webdav
```

Die Verzeichnisse `users`, `spool` und `webdav` müssen für den Benutzer beschreibbar sein, unter dem der Webserver läuft. Die restlichen Dateien müssen für diesen Benutzer lesbar sein. Die Benutzer- und Gruppennamen sind bei verschiedenen Distributionen unterschiedlich (z.B. bei Debian/Ubuntu `www-data`, bei Fedora `apache` oder bei openSUSE `wwwrun`).

Der folgende Befehl ändert den Besitzer für die oben genannten Verzeichnisse auf einem Debian/Ubuntu-System:

```
chown -R www-data users spool webdav
```

Weiterhin muss der Webserver-Benutzer in den Verzeichnissen `templates` und `users` Unterverzeichnisse für jeden neuen Benutzer anlegen dürfen, der in kivitendo angelegt wird:

```
chown www-data templates
```

2.5. kivitendo-Konfigurationsdatei

2.5.1. Einführung

In kivitendo gibt es nur noch eine Konfigurationsdatei, die benötigt wird: `config/kivitendo.conf` (kurz: "die Hauptkonfigurationsdatei"). Diese muss bei der Erstinstallation von kivitendo bzw. der Migration von älteren Versionen angelegt werden.

Als Vorlage dient die Datei `config/kivitendo.conf.default` (kurz: "die Default-Datei"):

```
$ cp config/kivitendo.conf.default config/kivitendo.conf
```

Die Default-Datei wird immer zuerst eingelesen. Werte, die in der Hauptkonfigurationsdatei stehen, überschreiben die Werte aus der Default-Datei. Die Hauptkonfigurationsdatei muss also nur die Abschnitte und Werte enthalten, die von denen der Default-Datei abweichen.

Anmerkung

Vor der Umbenennung in kivitendo hieß diese Datei noch `config/lx_office.conf`. Aus Gründen der Kompatibilität wird diese Datei eingelesen, sofern die Datei `config/kivitendo.conf` nicht existiert.

Diese Hauptkonfigurationsdatei ist dann eine installationsspezifische Datei, d.h. sie enthält bspw. lokale Passwörter und wird auch nicht im Versionsmanagement (git) verwaltet.

Die Konfiguration ist ferner serverabhängig, d.h. für alle Mandaten, bzw. Datenbanken gleich.

2.5.2. Abschnitte und Parameter

Die Konfigurationsdatei besteht aus mehreren Teilen, die entsprechend kommentiert sind:

- `authentication` (siehe Abschnitt "Abschnitt 2.10, „Benutzerauthentifizierung und Administratorpasswort“ [24]" in diesem Kapitel)
- `authentication/database`

- `authentication/http_basic`
- `authentication/http_headers`
- `authentication/ldap`
- `system`
- `paths`
- `mail_delivery` (siehe Abschnitt "E-Mail-Versand über einen SMTP-Server [30])
- `imap_client` (siehe Abschnitt "E-Mails in kivitendo importieren [31])
- `sent_emails_in_imap` (siehe Abschnitt "Versendete E-Mails über IMAP exportieren [31])
- `applications`
- `environment`
- `print_templates`
- `task_server`
- `periodic_invoices`
- `self_tests`
- `console`
- `testing`
- `testing/database`
- `debug`

Die üblicherweise wichtigsten Parameter, die am Anfang einzustellen oder zu kontrollieren sind, sind:

```
[authentication]
admin_password = geheim

[authentication/database]
host     = localhost
port     = 5432
db       = kivitendo_auth
user     = postgres
password =

[system]
default_manager = german
```

Für kivitendo Installationen in der Schweiz sollte hier `german` durch `swiss` ersetzt werden.

Die Einstellung `default_manager = swiss` bewirkt:

- Beim Erstellen einer neuen Datenbank in der kivitendo Administration werden automatisch die Standard-Werte für die Schweiz voreingestellt: Währung CHF, 5er-Rundung, Schweizer KMU-Kontenplan, Sollversteuerung, Aufwandsmethode, Bilanzierung (die Werte können aber manuell angepasst werden).
- Einstellen der Standardkonten für Rundungserträge und -aufwendungen (unter Mandantenkonfiguration → Standardkonten veränderbar)
- das verwendete Zahlenformat wird auf `1'000.00` eingestellt (unter Programm → Benutzereinstellungen veränderbar)

- DATEV-Automatik und UStVA werden nicht angezeigt, Erfolgsrechnung ersetzt GUV (unter Mandantenkonfiguration → Features veränderbar)

Nutzt man wiederkehrende Rechnungen, kann man unter `[periodic_invoices]` den Login eines Benutzers angeben, der nach Erstellung der Rechnungen eine entsprechende E-Mail mit Informationen über die erstellten Rechnungen bekommt.

kivitendo bringt eine eigene Komponente zur zeitgesteuerten Ausführung bestimmter Aufgaben mit, den Task-Server. Er wird u.a. für Features wie die wiederkehrenden Rechnungen benötigt, erledigt aber auch andere erforderliche Aufgaben und muss daher in Betrieb genommen werden. Seine Einrichtung wird im Abschnitt Task-Server genauer beschrieben.

Für Entwickler finden sich unter `[debug]` wichtige Funktionen, um die Fehlersuche zu erleichtern.

2.5.3. Versionen vor 2.6.3

In älteren kivitendo Versionen gab es im Verzeichnis `config` die Dateien `authentication.pl` und `lx-erp.conf`, die jeweils Perl-Dateien waren. Es gab auch die Möglichkeit, eine lokale Version der Konfigurationsdatei zu erstellen (`lx-erp-local.conf`). Dies ist ab 2.6.3 nicht mehr möglich, aber auch nicht mehr nötig.

Beim Update von einer kivitendo-Version vor 2.6.3 auf 2.6.3 oder jünger müssen die Einstellungen aus den alten Konfigurationsdateien manuell übertragen und die alten Konfigurationsdateien anschließend gelöscht oder verschoben werden. Ansonsten zeigt kivitendo eine entsprechende Fehlermeldung an.

2.6. Anpassung der PostgreSQL-Konfiguration

PostgreSQL muss auf verschiedene Weisen angepasst werden.

Dies variert je nach eingesetzter Distribution, da distributionsabhängig unterschiedliche Strategien beim Upgrade der Postgres Version eingesetzt werden. Als Hinweis einige Links zu den drei Distribution (Stand Dezember 2018):

- Fedora (Postgres-Installation unter Fedora)[4]
- Ubuntu (Infos für Postgres für die aktuelle LTS Version)[5]
- OpenSuSE (aktuell nur bis Version OpenSuSE 13 verifiziert)[6]

2.6.1. Änderungen an Konfigurationsdateien

In der Datei `postgresql.conf`, die je nach Distribution in verschiedenen Verzeichnissen liegen kann (z.B. `/var/lib/pgsql/data/` oder `/etc/postgresql/`), muss sichergestellt werden, dass TCP/IP-Verbindungen aktiviert sind. Das Verhalten wird über den Parameter `listen_address` gesteuert. Laufen PostgreSQL und kivitendo auf demselben Rechner, so kann dort der Wert `localhost` verwendet werden. Andernfalls müssen Datenbankverbindungen auch von anderen Rechnern aus zugelassen werden, was mit dem Wert `*` geschieht.

In der Datei `pg_hba.conf`, die im gleichen Verzeichnis wie die `postgresql.conf` zu finden sein sollte, müssen die Berechtigungen für den Zugriff geändert werden. Hier gibt es mehrere Möglichkeiten. Sinnvoll ist es nur die nötigen Verbindungen immer zuzulassen, für eine lokal laufende Datenbank zum Beispiel:

```
local all kivitendo password
host all kivitendo 127.0.0.1 255.255.255.255 scram-sha-256
```

2.6.2. Erweiterung für servergespeicherte Prozeduren

In der Datenbank `template1` muss die Unterstützung für servergespeicherte Prozeduren eingerichet werden. Melden Sie sich dafür als Benutzer “postgres” an der Datenbank an:

[4] https://fedoraproject.org/wiki/PostgreSQL
[5] https://help.ubuntu.com/lts/serverguide/postgresql.html
[6] https://de.opensuse.org/PostgreSQL

```
su - postgres
psql template1
```

führen Sie die folgenden Kommandos aus:

```
CREATE EXTENSION IF NOT EXISTS plpgsql;
\q
```

Anmerkung

`CREATE EXTENSION` ist seit Version 9.1 die bevorzugte Syntax um die Sprache `plpgsql` anzulegen. In diesen Versionen ist die Extension meist auch schon vorhanden. Sollten Sie eine ältere Version von Postgres haben, benutzen Sie stattdessen den folgenden Befehl.

```
CREATE LANGUAGE 'plpgsql';
\q
```

2.6.3. Erweiterung für Trigram Prozeduren

Ab Version 3.5.1 wird die Trigram-Index-Erweiterung benötigt. Diese wird mit dem SQL-Updatescript sql/Pg-upgrade2/trigram_extension.sql und Datenbank-Super-Benutzer Rechten automatisch installiert. Dazu braucht der DatenbankSuperbenutzer "postgres" ein Passwort.

```
su - postgres
psql
\password postgres

Eingabe Passwort
\q
```

Benutzername Postgres und Passwort können jetzt beim Anlegen einer Datenbank bzw. bei Updatescripten, die Superuser-Rechte benötigen, eingegeben werden.

Anmerkung

`pg_trgm` ist je nach Distribution nicht im Standard-Paket von Postgres enthalten. Ein

```
select * from pg_available_extensions where name ='pg_trgm';
```

in template1 sollte entsprechend erfolgreich sein. Andernfalls muss das Paket nachinstalliert werden, bspw. bei debian/ubuntu

```
apt install postgresql-contrib
```

2.6.4. Datenbankbenutzer anlegen

Wenn Sie nicht den Datenbanksuperuser “postgres” zum Zugriff benutzen wollen, so sollten Sie bei PostgreSQL einen neuen Benutzer anlegen. Ein Beispiel, wie Sie einen neuen Benutzer anlegen können:

Die Frage, ob der neue User Superuser sein soll, können Sie mit nein beantworten, genauso ist die Berechtigung neue User (Roles) zu generieren nicht nötig.

```
su - postgres
createuser -d -P kivitendo
exit
```

Wenn Sie später einen Datenbankzugriff konfigurieren, verändern Sie den evtl. voreingestellten Benutzer “postgres” auf “kivitendo” bzw. den hier gewählten Benutzernamen.

2.7. Webserver-Konfiguration

In diesem Abschnitt wird die Konfiguration des Apache-Webservers beschrieben. kivitendo wird mittels FastCGI/FCGI ausgeführt.

Es ist empfehlenswert, SSL einzusetzen, um die Daten per HTTPS verschlüsselt zwischen Browser und Webserver über das Netzwerk zu übertragen. Eine Möglichkeit dazu ist das Erstellen eines self-signed SSL Zertifikates, was unter Debian/Ubuntu durch Installieren des Pakets `ssl-cert` geschehen kann.

Der Zugriff auf den Installationspfad von kivitendo im Dateisystem muss in der Apache Webserverkonfigurationsdatei eingestellt werden, welche beim Starten des Webservers eingelesen wird. Wird SSL/HTTPS eingesetzt, so ist dies die Datei `default-ssl.conf`, für unverschlüsseltes HTTP ist es die Datei `000-default.conf`.

Bitte konsultieren Sie die Dokumentation des Apache-Webservers und Ihres Betriebssystems. Es kann erforderlich sein, das SSL-Modul und die Webserverkonfigurationsdatei zu aktivieren:

```
a2enmod ssl
a2ensite default-ssl
```

Unter Fedora und openSUSE müssen weiterhin in der Firewall die Ports 80 (HTTP) bzw. 443 (HTTPS) geöffnet werden.

2.7.1. Konfiguration für FastCGI/FCGI

Mit FastCGI wird der kivitendo-Programmcode beim Start des Webservers einmal geladen und danach wird nur die eigentliche Programmlogik ausgeführt.

Zuerst muss das FastCGI-Modul aktiviert werden. Dies kann unter Debian/Ubuntu z.B. mit folgendem Befehl geschehen:

```
a2enmod fcgid
```

Die Konfiguration für die Verwendung von kivitendo mit FastCGI erfolgt durch Einfügen von `Alias`- und `Directory`-Direktiven. Dabei wird zwischen dem Installationspfad von kivitendo im Dateisystem ("`/var/www/kivitendo-erp`") und der URL unterschieden, unter der kivitendo im Webbrowser erreichbar ist ("`/url/for/kivitendo-erp`").

Folgender Konfigurationsschnipsel funktioniert mit mod_fcgid:

```
AddHandler         fcgid-script .fcgi
FcgidMaxRequestLen 104857600
FcgidIOTimeout     600

AliasMatch ^/url/for/kivitendo-erp/[^/]+\.pl$ /var/www/kivitendo-erp/fcgi/dispatcher.fcgi
Alias       /url/for/kivitendo-erp/webdav     /var/www/kivitendo-erp/webdav
Alias       /url/for/kivitendo-erp            /var/www/kivitendo-erp/public

<Directory "/var/www/kivitendo-erp/fcgi">
  Options +ExecCGI
  Require all granted
</Directory>

<Directory "/var/www/kivitendo-erp/public">
  Options -ExecCGI +FollowSymlinks
  Require all granted
</Directory>

#
# Der folgende Abschnitt wird nur bei Verwendung des veralteten WebDAV
# Dateien-Backends benötigt, um die Links zu den Dokumenten in der Oberfläche
# aufrufen zu können.
```

```
#
Alias         /webdav                          /var/www/kivitendo-erp/webdav/
<Directory "/var/www/kivitendo-erp/webdav">
  Options -ExecCGI -FollowSymlinks
  Require all granted
</Directory>
```

Soll kivitendo direkt unter dem top-Level Pfad / des Webservers aufgerufen werden und nicht etwa unter dem Pfad /url/for/kivitendo-erp/, so muss entweder das DocumentRoot auf /var/www/kivitendo-erp/public gesetzt werden oder ein Alias mit Slash als Verzeichnisseparator nach public/ gesetzt werden. Folgendes Beispiel illustriert dies:

```
# alternativ zum zweiten Alias: DocumentRoot setzen
# DocumentRoot  /var/www/kivitendo-erp/public

AliasMatch ^/[^/]+\.pl$ /var/www/kivitendo-erp/fcgi/dispatcher.fcgi
Alias       /           /var/www/kivitendo-erp/public/
```

Hierdurch wird nur ein zentraler Dispatcher gestartet. Alle Zugriffe auf die einzelnen Scripte werden auf diesen umgeleitet. Dadurch, dass zur Laufzeit öfter mal Scripte neu geladen werden, gibt es hier kleine Performance-Einbußen.

Es ist durch die Webserver-Konfiguration sicherzustellen, dass insbesondere die Verzeichnisse config/, .git/ sowie die Verzeichnisse für DMS und Webdav nicht ausgeliefert werden, damit keine vertraulichen Daten (ohne Authentifizierung) preisgegeben werden. Es genügt, das Verzeichnis public/ per Webserver auszuliefern.

Insbesondere ist darauf zu achten, dass das DocumentRoot-Verzeichnis des Webservers keine vertraulichen Daten enthält, so dass auch im Falle einer Fehlkonfiguration des Webservers keine vertraulichen Daten leaken. Es empfiehlt sich, DocumentRoot entweder auf kivitendos public-Verzeichnis oder auf ein eigens dafür angelegtes leeres Verzeichnis zu setzen.

Seit mod_fcgid-Version 2.3.6 gelten sehr kleine Grenzen für die maximale Größe eines Requests. Mit folgender Zeile wird diese Grenze auf 100 MiB hochgesetzt:

```
FcgidMaxRequestLen 104857600
```

Standardmäßig wartet der Apache 40 Sekunden bevor es bei einer Anfrage zu einem Timeout und somit einenm 'Internal Server Error' kommt. Das ist für einige Berichte/Datenmenge etwas knapp und der Eintrag:

```
FcgidIOTimeout 600
```

setzt dieses Timeout auf 10 Minuten. Weitere Parameter sind in der Apache Dokumentation[7] einsehbar.

2.7.2. Aktivierung von mod_rewrite/directory_match für git basierte Installationen

Aufgrund von aktuellen (Mitte 2020) Sicherheitswarnungen für git basierte Webanwendungen ist die mitausgelieferte .htaccess restriktiver geworden und verhindert somit das Auslesen von git basierten Daten. Für debian/ubuntu muss das Modul mod_rewrite einmalig so aktiviert werden:

```
a2enmod rewrite
```

Alternativ und für Installationen ohne Apache ist folgender Artikel interessant: git-lücke[8]. Anstelle des dort beschriebenen DirectoryMatch für Apache2 würden wir etwas weitergehend auch noch das Verzeichnis config miteinbeziehen sowie ferner auch die Möglichkeit nicht ausschließen, dass es in Unterverzeichnissen auch noch .git Repositories geben kann. Die Empfehlung für Apache 2.4 wäre damit:

```
        <DirectoryMatch "/(\.git|config)/">
```

[7] https://httpd.apache.org/mod_fcgid/mod/mod_fcgid.html
[8] https://www.cyberscan.io/blog/git-luecke

```
    Require all denied
  </DirectoryMatch>
```

2.7.3. Weitergehende Konfiguration

Für einen deutlichen Sicherheitsmehrwert sorgt die Ausführung von kivitendo nur über https-verschlüsselten Verbindungen, sowie weiteren Zusatzmassnahmen, wie beispielsweise Basic Authentication. Die Konfigurationsmöglichkeiten sprengen allerdings den Rahmen dieser Anleitung, hier ein Hinweis auf einen entsprechenden Foreneintrag (Stand Sept. 2015)[9] und einen aktuellen (Stand Mai 2017) SSL-Konfigurations-Generator[10].

2.7.4. Aktivierung von Apache2 modsecurity

Aufgrund des OpenSource Charakters ist kivitendo nicht "out of the box" sicher. Organisatorisch empfehlen wir hier die enge Zusammenarbeit mit einem kivitendo Partner der auch in der Lage ist weiterführende Fragen in Bezug auf Datenschutz und Datensicherheit zu beantworten. Unabhängig davon empfehlen wir im Webserver Bereich die Aktivierung und Konfiguration des Moduls modsecurity für den Apache2, damit XSS und SQL-Injections verhindert werden.

Als Idee hierfür sei dieser Blog-Eintrag genannt: Test Apache2 modsecurity for SQL Injection[11].

2.8. Der Task-Server

Der Task-Server ist ein Prozess, der im Hintergrund läuft, in regelmäßigen Abständen nach abzuarbeitenden Aufgaben sucht und diese zu festgelegten Zeitpunkten abarbeitet (ähnlich wie Cron). Dieser Prozess wird u.a. für die Erzeugung der wiederkehrenden Rechnungen und weitere essenzielle Aufgaben benutzt.

Der Task-Server muss einmalig global in der Konfigurationsdatei konfiguriert werden. Danach wird er für jeden Mandanten, für den er laufen soll, in der Adminsitrationsmaske eingeschaltet.

Beachten Sie, dass der Task-Server in den Boot-Vorgang Ihres Servers integriert werden muss, damit er automatisch gestartet wird. Dies kann kivitendo nicht für Sie erledigen.

Da der Task-Server als Perlscript läuft, wird Arbeitsspeicher, der einmal benötigt wurde, nicht mehr an das Betriebssystem zurückgegeben, solange der Task-Server läuft. Dies kann dazu führen, dass ein länger laufender Task-Server mit der Zeit immer mehr Arbeitsspeicher für sich beansprucht. Es ist deshalb sinnvoll, dass der Task-Server in regelmässigen Abständen neu gestartet wird. Allerdings berücksichtigt der Task-Server ein Memory-Limit, wenn dieses in der Konfigurationsdatei angegeben ist. Bei Überschreiten dieses Limits beendet sich der Task-Server. Sofern der Task-Server als systemd-Service mit dem mitgelieferten Skript eingerichtet wurde, startet dieser danach automatisch erneut.

2.8.1. Verfügbare und notwendige Konfigurationsoptionen

Die Konfiguration erfolgt über den Abschnitt `[task_server]` in der Datei `config/kivitendo.conf`. Die dort verfügbaren Optionen sind:

`run_as`
: Wird der Server vom Systembenutzer `root` gestartet, so wechselt er auf den mit `run_as` angegebenen Systembenutzer. Der Systembenutzer muss dieselben Lese- und Schreibrechte haben, wie auch der Webserverbenutzer (siehe see Manuelle Installation des Programmpaketes [9]). Daher ist es erforderlich, hier denselben Systembenutzer einzutragen, unter dem auch der Webserver läuft.

`debug`
: Schaltet Debug-Informationen an und aus.

[9] https://www.kivitendo.de/redmine/boards/1/topics/142
[10] https://mozilla.github.io/server-side-tls/ssl-config-generator/
[11] https://doxsec.wordpress.com/2017/06/11/using-modsecurity-web-application-firewall-to-prevent-sql-injection-and-xss-using-blocking-rules/

2.8.2. Konfiguration der Mandanten für den Task-Server

Ist der Task-Server grundlegend konfiguriert, so muss anschließend jeder Mandant, für den der Task-Server laufen soll, einmalig konfiguriert werden. Dazu kann in der Maske zum Bearbeiten von Mandanten im Administrationsbereich eine kivitendo-Benutzerkennung ausgewählt werden, unter der der Task-Server seine Arbeit verrichtet.

Ist in dieser Einstellung keine Benutzerkennung ausgewählt, so wird der Task-Server für diesen Mandanten keine Aufgaben ausführen.

2.8.3. Automatisches Starten des Task-Servers beim Booten

Der Task-Server verhält sich von seinen Optionen her wie ein reguläres SystemV-kompatibles Boot-Script. Außerdem wechselt er beim Starten automatisch in das kivitendo-Installationsverzeichnis.

Deshalb ist es möglich, ihn durch Setzen eines symbolischen Links aus einem der Runlevel-Verzeichnisse heraus in den Boot-Prozess einzubinden. Da das bei neueren Linux-Distributionen aber nicht zwangsläufig funktioniert, werden auch Start-Scripte mitgeliefert, die anstelle eines symbolischen Links verwendet werden können.

2.8.3.1. SystemV-basierende Systeme (z.B. ältere Debian, ältere openSUSE, ältere Fedora)

Kopieren Sie die Datei `scripts/boot/system-v/kivitendo-task-server` nach `/etc/init.d/kivitendo-task-server`. Passen Sie in der kopierten Datei den Pfad zum Task-Server an (Zeile `DAEMON=....`). Binden Sie das Script in den Boot-Prozess ein. Dies ist distributionsabhängig:

- Debian-basierende Systeme:

  ```
  update-rc.d kivitendo-task-server defaults
  insserv kivitendo-task-server
  ```

- Ältere openSUSE und ältere Fedora:

  ```
  chkconfig --add kivitendo-task-server
  ```

Danach kann der Task-Server mit dem folgenden Befehl gestartet werden:

```
/etc/init.d/kivitendo-task-server start
```

2.8.3.2. Upstart-basierende Systeme (z.B. Ubuntu bis 14.04)

Kopieren Sie die Datei `scripts/boot/upstart/kivitendo-task-server.conf` nach `/etc/init/kivitendo-task-server.conf`. Passen Sie in der kopierten Datei den Pfad zum Task-Server an (Zeile `exec ....`).

Danach kann der Task-Server mit dem folgenden Befehl gestartet werden:

```
service kivitendo-task-server start
```

2.8.3.3. systemd-basierende Systeme (z.B. neure openSUSE, neuere Fedora, neuere Ubuntu und neuere Debians)

Kopieren Sie die Datei `scripts/boot/systemd/kivitendo-task-server.service` nach `/etc/systemd/system/`. Passen Sie in der kopierten Datei den Pfad zum Task-Server an (Zeilen `ExecStart=....` und `ExecStop=...`).

Machen Sie anschließend das Script systemd bekannt, und binden Sie es in den Boot-Prozess ein. Dazu führen Sie die folgenden Befehl aus:

```
systemctl daemon-reload
systemctl enable kivitendo-task-server.service
```

Wenn Sie den Task-Server jetzt sofort starten möchten, anstatt den Server neu zu starten, so können Sie das mit dem folgenden Befehl tun:

```
systemctl start kivitendo-task-server.service
```

Ein so eingerichteter Task-Server startet nach Beendigung automatisch erneut. Das betrifft eine Beendigung über die Oberfläche, eine Beendingung über die Prozesskontrolle und eine Beendigung bei Überschreiten des Memory-Limits. Soll der Task-Server nicht erneut starten, so können Sie ihn mit folgendem Befehl stoppen:

```
systemctl stop kivitendo-task-server.service
```

2.8.4. Wie der Task-Server gestartet und beendet wird

Der Task-Server wird wie folgt kontrolliert:

```
./scripts/task_server.pl Befehl
```

`Befehl` ist dabei eine der folgenden Optionen:

- `start` startet eine neue Instanz des Task-Servers. Die Prozess-ID wird innerhalb des `users`-Verzeichnisses abgelegt.
- `stop` beendet einen laufenden Task-Server.
- `restart` beendet und startet ihn neu.
- `status` berichtet, ob der Task-Server läuft.

Der Task-Server wechselt beim Starten automatisch in das kivitendo-Installationsverzeichnis.

Dieselben Optionen können auch für die SystemV-basierenden Runlevel-Scripte benutzt werden (siehe oben).

Wurde der Task-Server als systemd-Service eingerichtet (s.o.), so startet dieser nach Beendigung automatisch erneut.

2.9. Konfiguration der Hintergrund-Jobs

Hintergrund-Jobs werden über System -> Hintergrund-Jobs und Task-Server -> Aktuelle Hintergrund-Jobs anzeigen -> Aktions-Knopf 'erfassen' angelegt.

Nachdem wir über das Menü dort angelangt sind, legen wir hier unseren Hintergrund-Jobs an:

- `Aktiv:` Hier ein 'Ja' auswählen
- `Ausführungsart:` 'wiederholte Ausführung' auswählen
- `Paketname:` Hintergrundjob auswählen
- `Ausführungszeitplan:` Hier entsprechend Werte wie in der crontab eingeben.

  Syntax:

  ```
  * * * * *
  # # # # #
  # # # # #
  # # # # ##### Wochentag (0-7, Sonntag ist 0 oder 7)
  # # # ####### Monat (1-12)
  # # ######### Tag (1-31)
  # ########### Stunde (0-23)
  ```

```
############# Minute (0-59)
```

Die Sterne können folgende Werte haben:

```
1 2 3 4 5

1 = Minute (0-59)
2 = Stunde (0-23)
3 = Tag (0-31)
4 = Monat (1-12)
5 = Wochentag (0-7, Sonntag ist 0 oder 7)
```

Um die Ausführung auf eine Minute vor den Jahreswechsel zu setzen, müssen die folgenden Werte eingetragen werden:

```
59 23 31 12 *
```

- `Daten:` In diesem Feld können optionale Parameter für den Hintergrund im YAML-Format gesetzt werden.

2.9.1. SetNumberRange

Der Hintergrund-Job `SetNumberRange` kann entweder jährlich oder monatlich/täglich den Nummernkreis verändern. Der Standardmodus 'jährlich' kodiert die nächste Jahreszahl in alle Nummernkreise und initialisiert diese von Beginn, bpsw. 202500001. Zusätzlich bleiben Präfixe in den Nummernkreis erhalten und es gibt drei konfigurierbare Parameter in diesem Modus:

Der jährliche Modus (default) akzeptiert im Feld `Daten` drei optionale Parameter, nämlich `digit_year`, `multiplier` so wie `current_year`.

`digit_year` kann zwei Werte haben entweder 2 oder 4, darüber wird gesteuert ob die Jahreszahl zwei oder vierstellig kodiert wird (für 2019, dann entweder 19 oder 2019). Der Standardwert ist vierstellig.

`multiplier` ist ein Vielfaches von 10, darüber wird die erste Nummer im Nummernkreis (die Anzahl der Stellen) wie folgt bestimmt:

```
multiplier     Nummernkreis 2020
10        ->   20200
100       ->   202000
1000      ->   2020000
```

Falls der Parameter `current_year` bspw. so gesetzt ist:

```
current_year: 1
```

wird der Nummernkreis nicht um eins hochgezählt. Das ist sinnvoll wenn vom 31.12. auf den 01.01. sowieso keine Rechnungsläufe stattfinden und man die Nummernkreise dann am 01.01. des neuen Jahres automatisch hochsetzen möchte.

Wir gehen jetzt beispielhaft von einer letzten Rechnungsnummer von RE2019456 aus. Demnach sollte ab Januar 2020 die erste Nummer RE2020001 sein. Da der Task auch Präfixe berücksichtigt, kann dies mit folgenden JSON-kodierten Werten umgesetzt werden:

```
Daten:

multiplier: 100
digits_year: 4
```

Dieser Job müsste dann zwingend vor Mitternacht des 31.12. ausgeführt werden.

```
Daten:
```

```
multiplier: 100
digits_year: 4
current_year: 1
```

Mit dieser Einstellung kann der Job auf 00:01h des 01.01. gesetzt werden.

Der (wahrscheinliche) monatliche Modus wird über den Parameter `monthly` gesteuert und erzeugt standardmäßig folgende Nummernkreise JJ-MM-000, bspw. am 01.04.2025 25-04-000. Dieser Modus akzeptiert die zwei optionalen Parameter `monthly_postfix` für die Überlagerung am Ende der Zeichenkette und `monthly_strftime` für die beliebige Formatierung der strftime C-Methode, s.a. strftime-Patterns[12]. Folgende Parameter erzeugen am 01.04.2025 folgenden Nummernkreis: 2025-04-01-A0001.

```
Daten:


monthly: 1
monthly_postfix: '-A0001'
monthly_strftime: '%Y-%m-%d'
```

2.9.2. ImportRecordEmails

Die prinzipielle Idee des Hintergrund-Jobs `ImportRecordEmails` ist es auf einen spezifischen IMAP-Ordner zuzugreifen und unterhalb diesen Ordners alle E-Mails in kivitendo zu importieren. Der Job ist transaktionssicher, d.h. entweder können alle E-Mails zum Zeitpunkt des Jobs importiert werden oder es wird überhaupt keine importiert.

Die eindeutige Kennzeichnung um Duplikate beim Import zu vermeiden ist die UID der E-Mail auf dem IMAP-Server, dies ist solange stabil, solange der Provider nicht gewechselt wird, bzw. die E-Mails nicht auf einen anderen IMAP-Server migrieren.

Zu guter Letzt kann der Job auch noch automatisch Verarbeitungen mit dem Anhängen der E-Mail direkt beim Import durchführen, bspw. ZUGFeRD konforme PDFs automatisch verbuchen.

Die prinzipiellen IMAP-Zugangsdaten können vollständig über das Feld Daten konfiguriert werden und sind wie folgt definiert:

- `hostname:` Hier wird der E-Mail-Server (IMAP) eingetragen
- `username:` Benutzername, für den IMAP-Server (häufig die E-Mail-Adresse)
- `password_secret:` Passwort des Benutzers als kivitendo Secrets (Siehe System -> Secrets)
- `password:` Passwort des Benutzers (deprecated, password_secret sollte benutzt werden)
- `folder:` Hier wird der Ordner eingetragen, aus dem die E-Mails importert werden sollen, bspw. 'INBOX'
- `port:` Port am E-Mail-Server. Default ist 993
- `ssl:` Gibt an ob SSL verwendet werden soll. Default: 1

Um das ganze Verfahren komfortabler zu optimieren oder Fehlimporte wieder zu löschen stehen folgende optionale Parameter zu Verfügung:

- `email_import_ids_to_delete:` Hier können IDs von Importen eingetragen werden, deren E-Mails aus dem E-Mail-Journal gelöscht werden sollen.
- `process_imported_emails:` Wenn nach dem Import noch weitere Verarbeitung der angehangenen Dokumente erfolgen soll, müssen hier die jeweiligen Schritte eingetragen werden. Aktuell ist es möglich, dass angehangene ZUGFeRD-Rechnung direkt verbucht und mit der E-Mail verknüpft werden. Dazu muss hier 'zugferd' eingetragen werden.

[12] https://metacpan.org/pod/DateTime#strftime-Patterns

- `imported_imap_flag:` Das hier eingetragenen Flag wird nach dem Verarbeiten in der E-Mail auf den IMAP-Server gesetzt.
- `not_imported_imap_flag:` Das hier eingetragenen Flag wird bei erfolglosen Importieren auf dem IMAP-Server gesetzt und ist hilfreich um die Fehlerfälle bei großem Datenimporten schneller einzugrenzen.
- `processed_imap_flag:` Das hier eingetragenen Flag wird nach dem erfolgreichen Verarbeiten an der E-Mail gesetzt.
- `not_processed_imap_flag:` Dieses Flag wird gesetzt, falls die E-Mail nicht erfolgreich verarbeitet werden konnte (Verarbeitungsprozeß ist in `process_imported_emails` definiert.
- `record_type:` Einträge im E-Mail-Journal können direkt einem Belegtypen zugeordnet werden. Wenn alle E-Mails, die mit einem Hintergrundjob importiert werden, den gleichen Belegtypen haben, kann man diesen hier festlegen und alle Einträge im E-Mail-Journal werden entsprechend zugeordnet. Für Eingangsrechnungen kann man hier bspw. 'ap_transaction' setzen.
- `skip_broken_mime_mails:` Bricht den Import-Lauf NICHT ab, falls einige Mails nicht MIME-konform "zerlegt" werden können. Default: 0

Wie die IMAP Flags von den jeweiligen Clients angezeigt und eingerichtet werden, ist aktuell im Thunderbird (Version 115.8.0 und Version 115.8.1) und SoGo (Version 5.9.1) getestet:

In Thunderbird heißen die Flags Schlagwörter. In unseren beiden Testfälle, war das Verfahren unterschiedlich:

Thunderbird 115.8.0: Sie werden durchnummeriert mit dem Prefix "$label". Über die Einstellungen kann man Schlagwort und Farbe für den jeweiligen Tag setzen und berabeiten. Um die vordefenierten Tags in Thunderbird zu nutzen kann man $label1 - $label5 nutzen. Eigene Labels werden dann von thunderbird automatisch hochgezählt. Um das korrekte Tag für ein Label zu finden oder auch selbst ein Tag mit einer selbst gewählten Zahl zu definieren kann man in den Einstellunge ganz am Ende über den Button `Konfiguration berabeiten...` die Werte in der Kofiguration einsehen, ändern und berabeiten. Hier muss man nach `mailnews.tags` suchen.

Thunderbird 115.8.1: Einstellungen -> Schlagwörter -> hinzufügen. Das Schlagwort wird mit dem 'Label' 'name_mit_unterstrichen' zu Verfügung gestellt. Was wirklich passiert kann man dann ganz unten in den Einstellungen unter 'Konfiguration bearbeiten' und einer darauf folgenden Filtersuche nach 'mailnews.tag' erkennen.

In SoGo kann man unter Einstellungen -> Mail -> Labels beliebige Label mit $ als Prefix anlegen und Namen und Farbe zuweisen.

Anmerkung

Hilfreich ist dieses Skript was die Flags beispielhaft für einen IMAP-Ordner 'Eingangsrechung' ausliest. Entsprechend die `$GROSSBUCHSTABEN` mit eigenen Werten ersetzen:

```
#!/usr/bin/perl

use strict;
use warnings;

use Mail::IMAPClient;
my $imap = Mail::IMAPClient->new(
  Server   => '$SERVER',
  User     => '$USER',
  Password => '$PASS',
  Ssl      => 1,
  Uid      => 1,
);

$imap->select("INBOX.Eingangsrechnung")
  or die "Select  error: ", $imap->LastError, "\n";
```

```
# get the flags for every message in my 'Eingangsrechner' folder
my $flaghash = $imap->flags( scalar( $imap->search("ALL") ) );

# pump through sorted hash keys to print results:
for my $k (keys %$flaghash) {
    # print: Message 1: \Flag1, \Flag2, \Flag3
    print "Message $k:\t", join( ", ", @{$flaghash->{$k}} ), "\n";
}
```

Anmerkung

Eine beispielhafte Konfiguration im YAML-Format für das Feld 'Daten' im Hintergrund-Job könnte bspw. so aussehen. Enthalten sind alle aktuell bekannten Parameter mit entsprechenden sinnvollen Standardwerten, bis auf die Werte der IMAP-Flags (s.a. vorherige Notiz), sodass die Konfiguration als einfachen Start zum Experimentieren direkt kopiert werden kann:

```
record_type: ap_transaction
folder: INBOX/Eingangsrechnung
imported_imap_flag: in_kivitendo_importiert
not_imported_imap_flag: nicht_importiert
process_imported_emails: zugferd
processed_imap_flag: zugferd_verarbeitet
not_processed_imap_flag: zugferd_geht_net
skip_broken_mime_mails: 0
hostname:  www.meine-domaene.de
username: alpha39@meine-domaene.de
password: supipass8
```

2.9.3. InventoryClearAll

Besteht die Anforderung, vor einer Inventur sämtliche Lagerplätze mittels Korrekturbuchungen auf Bestand 0 zu bringen, so kann dies mit dem Hintergrund-Job `InventoryClearAll` erledigt werden.

Anmerkung

Damit diese Korrekturbuchungen nur einmalig auf Nutzerwunsch ausgeführt werden und nicht ausversehen periodisch das Lager leer räumen, muss die Variable `correction_date` auf das aktuelle Datum gesetzt sein. Die Korrekturbuchungen können nicht ohne Weiteres rückgängig gemacht werden. Nachdem der Job im Menü System -> Hintergrund-Jobs und Task-Server -> Aktuelle Hintergrund-Jobs anzeigen erfasst wurde, kann er einmalig ausgeführt werden, indem beim Bearbeiten der Button `Speichern und Ausführen` gewählt wird.

Im Feld Daten werden folgende Variablen akzeptiert:

- `correction_date:` Das aktuelle Datum in der Form `YYYY-MM-DD`, sonst läuft der Job nicht
- `comment:` (optional) Wird in den Korrekturbuchungen als Kommentar gesetzt. Falls kein Wert gegeben ist, wird der Kommentar `vor Inventur` gesetzt
- `dry_run:` (optional) Falls `dry_run: 1` gesetzt ist, werden keine Korrekturbuchungen ausgeführt, die notwenigen Korrekturbuchungen werden lediglich im Ergebnis des Jobs angezeigt
- `warehouse:` (optional) Schränkt die Korrekturbuchungen auf ein bestimmtes Lager ein; andernfalls werden Korrekturbuchungen für alle Lager ausgeführt

Ein Konfigurationsbeispiel ist:

```
correction_date: 2025-03-11
comment: vor Inventur
dry_run: 1
# warehouse: Stahllager Industriepark Gallin
```

2.9.4. CloseQuotations

Gibt es viele alte offene Preisanfragen im Einkauf und Angebote im Verkauf, so können diese komfortabel mit dem Hintergrund-Job `CloseQuotations` geschlossen werden.

Im Feld Daten werden folgende Variablen akzeptiert:

- `dry_run:` (optional) Falls `dry_run: 1` gesetzt ist, werden keine Belege geschlossen, die Nummern der betroffenen Belege werden lediglich im Ergebnis des Jobs angezeigt
- `years:` (optional) Anzahl der Jahre, die offene Preisanfragen und Angebote alt sein müssen, damit sie geschlossen werden. Standardwert: `years: 1` (ein Jahr)

Ein Konfigurationsbeispiel ist:

```
dry_run: 1
years: 1
```

2.10. Benutzerauthentifizierung und Administratorpasswort

Informationen über die Einrichtung der Benutzerauthentifizierung, über die Verwaltung von Gruppen und weitere Einstellungen

2.10.1. Grundlagen zur Benutzerauthentifizierung

kivitendo verwaltet die Benutzerinformationen in einer Datenbank, die im folgenden “Authentifizierungsdatenbank” genannt wird. Für jeden Benutzer kann dort eine eigene Datenbank für die eigentlichen Finanzdaten hinterlegt sein. Diese beiden Datenbanken können, müssen aber nicht unterschiedlich sein.

Im einfachsten Fall gibt es für kivitendo nur eine einzige Datenbank, in der sowohl die Benutzerinformationen als auch die Daten abgelegt werden.

Zusätzlich ermöglicht es kivitendo, dass die Benutzerpasswörter gegen verschiedene Authentifizierungsmethoden geprüft werden. Dazu zählen die Authentifizierungsdatenbank, LDAP-Server sowie verschiedene Arten von HTTP-Header-basierten Methoden.

Welche Art der Passwortüberprüfung kivitendo benutzt und wie kivitendo die Authentifizierungsdatenbank erreichen kann, wird in der Konfigurationsdatei `config/kivitendo.conf` festgelegt. Diese muss bei der Installation und bei einem Upgrade von einer Version vor v2.6.0 angelegt werden. Eine Beispielkonfigurationsdatei `config/kivitendo.conf.default` existiert, die als Vorlage benutzt werden kann.

2.10.2. Administratorpasswort

Das Passwort, das zum Zugriff auf das Administrationsinterface von kivitendo benutzt wird, wird ebenfalls in dieser Datei gespeichert. Es kann auch nur dort und nicht mehr im Administrationsinterface selber geändert werden. Der Parameter dazu heißt `admin_password` im Abschnitt `[authentication]`.

2.10.3. Authentifizierungsdatenbank

Die Verbindung zur Authentifizierungsdatenbank wird mit den Parametern in `[authentication/database]` konfiguriert. Hier sind die folgenden Parameter anzugeben:

`host`
: Der Rechnername oder die IP-Adresse des Datenbankservers

`port`
: Die Portnummer des Datenbankservers, meist 5432

`db`
: Der Name der Authentifizierungsdatenbank

`user`
: Der Benutzername, mit dem sich kivitendo beim Datenbankserver anmeldet (z.B. `"postgres"`)

`password`
: Das Passwort für den Datenbankbenutzer

Die Datenbank muss noch nicht existieren. kivitendo kann sie automatisch anlegen (mehr dazu siehe unten).

2.10.4. Passwortüberprüfung

kivitendo unterstützt verschiedene Module für die Passwortüberprüfung. Welche benutzt wird, regelt der Parameter `module` im Abschnitt `[authentication]`.

Dieser Parameter listet die zu verwendenden Authentifizierungsmodule auf. Es muss mindestens ein Modul angegeben werden, es können aber auch mehrere angegeben werden. Weiterhin ist es möglich, das LDAP-Modul mehrfach zu verwenden und für jede Verwendung eine unterschiedliche Konfiguration zu nutzen, z.B. um einen Fallback-Server anzugeben, der benutzt wird, sofern der Hauptserver nicht erreichbar ist.

Verfügbare Module sind:

- `DB`: in Authentifizierungsdatenbank integrierte Benutzerverwaltung
- `ldap`: Bind mit User-Objekten gegen einen oder mehrere LDAP-Server
- `http_headers`: überlässt Authenfizierung vorgelagerten Proxy-Servern übernimmt Usernamen aus mitgeschickten HTTP-Headern

2.10.4.1. Authentifizierungsdatenbank

Sollen die Benutzerpasswörter in der Authentifizierungsdatenbank geprüft werden, so muss der Parameter `module` das Modul `DB` enthalten. Sofern das Modul in der Liste enthalten ist, egal an welcher Position, können sowohl der Administrator als auch die Benutzer selber ihre Passwörter in kivitendo ändern.

2.10.4.2. LDAP-Server

Wenn Passwörter gegen einen oder mehrere externe LDAP- oder Active-Directory-Server geprüft werden, so muss der Parameter `module` den Wert `LDAP` enthalten. In diesem Fall müssen zusätzliche Informationen über den LDAP-Server im Abschnitt `[authentication/ldap]` angegeben werden. Das Modul kann auch mehrfach angegeben werden, wobei jedes Modul eine eigene Konfiguration bekommen sollte. Der Name der Konfiguration wird dabei mit einem Doppelpunkt getrennt an den Modulnamen angehängt (`LDAP:Name-der-Konfiguration`). Der entsprechende Abschnitt in der Konfigurationsdatei lautet dann `[authentication/Name-der-Konfiguration]`.

Die verfügbaren Parameter für die LDAP-Konfiguration lauten:

`host`
: Der Rechnername oder die IP-Adresse des LDAP- oder Active-Directory-Servers. Diese Angabe ist zwingend erforderlich.

`port`
: Die Portnummer des LDAP-Servers; meist 389.

`tls`
: Wenn Verbindungsverschlüsselung gewünscht ist, so diesen Wert auf '`1`' setzen, andernfalls auf '`0`' belassen

`verify`
: Wenn Verbindungsverschlüsselung gewünscht und der Parameter *`tls`* gesetzt ist, so gibt dieser Parameter an, ob das Serverzertifikat auf Gültigkeit geprüft wird. Mögliche Werte sind `require` (Zertifikat wird überprüft und muss gültig sei; dies ist der Standard) und `none` (Zertifikat wird nicht überpfüft).

`attribute`
: Das LDAP-Attribut, in dem der Benutzername steht, den der Benutzer eingegeben hat. Für Active-Directory-Server ist dies meist '`sAMAccountName`', für andere LDAP-Server hingegen '`uid`'. Diese Angabe ist zwingend erforderlich.

`base_dn`
: Der Abschnitt des LDAP-Baumes, der durchsucht werden soll. Diese Angabe ist zwingend erforderlich.

`filter`
: Ein optionaler LDAP-Filter. Enthält dieser Filter das Wort `<%login%>`, so wird dieses durch den vom Benutzer eingegebenen Benutzernamen ersetzt. Andernfalls wird der LDAP-Baum nach einem Element durchsucht, bei dem das oben angegebene Attribut mit dem Benutzernamen identisch ist.

`bind_dn` und `bind_password`
: Wenn der LDAP-Server eine Anmeldung erfordert, bevor er durchsucht werden kann (z.B. ist dies bei Active-Directory-Servern der Fall), so kann diese hier angegeben werden. Für Active-Directory-Server kann als '`bind_dn`' entweder eine komplette LDAP-DN wie z.B. '`cn=Martin Mustermann,cn=Users,dc=firmendomain`' auch nur der volle Name des Benutzers eingegeben werden; in diesem Beispiel also '`Martin Mustermann`'.

`timeout`
: Timeout beim Verbindungsversuch, bevor der Server als nicht erreichbar gilt; Standardwert: 10

2.10.4.3. HTTP-Header: Username in Header

Diese Methode der Authentifizierung überlässt einem vorgelagerten Webserver oder Proxyserver die Authentifizierung. Dazu können SSO-Systeme wie z.B. Authelia oder Authentik zum Einsatz kommen. Die vorgelagerten Server übermitteln dann den Usernamen der authentifizierten Person in einem speziellen HTTP-Header, der an kivitendo durchgereicht wird. Damit kivitendo nicht von beliebigen Quellen aus mit so einem Usernamen aufgerufen werden kann, verlangt kivitendo weiterhin, dass in einem weiteren Header ein Shared Secret übertragen wird. Dieses Secret wird vom Administrator vergeben und sollte nicht weitergegeben werden.

Über einen weiteren Header wird wiederum gesteuert, an welchem Mandanten die Anmeldung erfolgt. Dies geschieht über die Datenbank-ID des Mandanten.

Um diese Methode zu aktivieren, muss das Authentifizierungsmodul auf `HTTPHeaders` gestellt werden. Zusätzlich muss im Abschnitt `authentication/http_headers` der Parameter `enabled=1` und im Abschnitt `authentication/http_basic` der Parameter `enabled=0` gesetzt werden. Die folgenden Parameter müssen anschließend im Abschnitt `authentication/http_headers` konfiguriert werden:

`client_id_header`
: Name des Headers, in dem das vorgelagerte System die Datenbank-ID des Mandanten überträgt.

`user_header`
: Name des Headers, in dem das vorgelagerte System den Namen des authentifizierten Users überträgt.

`secret_header`
: Name des Headers, in dem das vorgelagerte System das Shared Secret.

`secret`
: Wert des Shared Secrets selber, der vom vorgelagerten System übertragen werden muss, damit die Werte als gültig angesehen werden.

2.10.4.4. HTTP-Header: Basic Authorization

Diese Methode der Authentifizierung überlässt dem Webserver (meist Apache) die Authentifizierung mittels der standardisierten HTTP Basic Authentication (RFC 7617). Dazu muss der Webserver so konfiguriert werden, dass er für alle kivitendo-Requests vom Webbrowser Authentifizierung verlangt. Zusätzlich muss er veranlasst werden, den angegebenen Usernamen auch an kivitendo zu übermitteln. Dies könnte beispielhaft wie folgt aussehen:

```
<Directory "/var/www/kivitendo-erp/fcgi">
  Options +ExecCGI

  # Require authentication from local user file:
  AuthType Basic
  AuthName "kivitendo"
  AuthUserFile /etc/apache2/htpasswd.kivitendo
  Require valid-user

  # Pass name of authorized user to kivitendo:
  SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
</Directory>

<DirectoryMatch "^/var/www/kivitendo-erp/(public|webdav)">
  AuthType Basic
  AuthName "kivitendo"
  AuthUserFile htpasswd.kivitendo
  Require valid-user
</DirectoryMatch>
```

Um diese Methode zu aktivieren, muss das Authentifizierungsmodul auf `HTTPHeaders` gestellt werden. Zusätzlich muss im Abschnitt `authentication/http_basic` der Parameter `enabled=1` und im Abschnitt `authentication/http_headers` der Parameter `enabled=0` gesetzt werden.

2.10.5. Name des Session-Cookies

Sollen auf einem Server mehrere kivitendo-Installationen aufgesetzt werden, so müssen die Namen der Session-Cookies für alle Installationen unterschiedlich sein. Der Name des Cookies wird mit dem Parameter `cookie_name` im Abschnitt `[authentication]`gesetzt.

Diese Angabe ist optional, wenn nur eine Installation auf dem Server existiert.

2.10.6. Anlegen der Authentifizierungsdatenbank

Nachdem alle Einstellungen in `config/kivitendo.conf` vorgenommen wurden, muss kivitendo die Authentifizierungsdatenbank anlegen. Dieses geschieht automatisch, wenn Sie sich im Administrationsmodul anmelden, das unter der folgenden URL erreichbar sein sollte:

http://localhost/kivitendo-erp/controller.pl?action=Admin/login

2.11. Mandanten-, Benutzer- und Gruppenverwaltung

Nach der Installation müssen Mandanten, Benutzer, Gruppen und Datenbanken angelegt werden. Dieses geschieht im Administrationsmenü, das Sie unter folgender URL finden:

http://localhost/kivitendo-erp/controller.pl?action=Admin/login

Verwenden Sie zur Anmeldung das Passwort, das Sie in der Datei `config/kivitendo.conf` eingetragen haben.

2.11.1. Zusammenhänge

kivitendo verwaltet zwei Sets von Daten, die je nach Einrichtung in einer oder zwei Datenbanken gespeichert werden.

Das erste Set besteht aus Anmeldeinformationen: welche Benutzer und Mandanten gibt es, welche Gruppen, welche BenutzerIn hat Zugriff auf welche Mandanten, und welche Gruppe verfügt über welche Rechte. Diese Informationen werden in der Authentifizierungsdatenbank gespeichert. Dies ist diejenige Datenbank, deren Verbindungsparameter in der Konfigurationsdatei `config/kivitendo.conf` gespeichert werden.

Das zweite Set besteht aus den eigentlichen Verkehrsdaten eines Mandanten, wie beispielsweise die Stammdaten (Kunden, Lieferanten, Waren) und Belege (Angebote, Lieferscheine, Rechnungen). Diese werden in einer Mandantendatenbank gespeichert. Die Verbindungsinformationen einer solchen Mandantendatenbank werden im Administrationsbereich konfiguriert, indem man einen Mandanten anlegt und dort die Parameter einträgt. Dabei hat jeder Mandant eine eigene Datenbank.

Aufgrund des Datenbankdesigns ist es für einfache Fälle möglich, die Authentifizierungsdatenbank und eine der Mandantendatenbanken in ein und derselben Datenbank zu speichern. Arbeitet man hingegen mit mehr als einem Mandanten, wird empfohlen, für die Authentifizierungsdatenbank eine eigene Datenbank zu verwenden, die nicht gleichzeitig für einen Mandanten verwendet wird.

2.11.2. Mandanten, Benutzer und Gruppen

kivitendos Administration kennt Mandanten, Benutzer und Gruppen, die sich frei zueinander zuordnen lassen.

kivitendo kann mehrere Mandaten aus einer Installation heraus verwalten. Welcher Mandant benutzt wird, kann direkt beim Login ausgewählt werden. Für jeden Mandanten wird ein eindeutiger Name vergeben, der beim Login angezeigt wird. Weiterhin benötigt der Mandant Datenbankverbindungsparameter für seine Mandantendatenbank. Diese sollte über die Datenbankverwaltung geschehen.

Ein Benutzer ist eine Person, die Zugriff auf kivitendo erhalten soll. Sie erhält einen Loginnamen sowie ein Passwort. Weiterhin legt der Administrator fest, an welchen Mandanten sich ein Benutzer anmelden kann, was beim Login verifiziert wird.

Gruppen dienen dazu, Benutzern innerhalb eines Mandanten Zugriff auf bestimmte Funktionen zu geben. Einer Gruppe werden dafür vom Administrator gewisse Rechte zugeordnet. Weiterhin legt der Administrator fest, für welche Mandanten eine Gruppe gilt, und welche Benutzer Mitglieder in dieser Gruppe sind. Meldet sich ein Benutzer dann an einem Mandanten an, so erhält er alle Rechte von allen denjenigen Gruppen, die zum Einen dem Mandanten zugeordnet sind und in denen der Benutzer zum Anderen Mitglied ist,

Die Reihenfolge, in der Datenbanken, Mandanten, Gruppen und Benutzer angelegt werden, kann im Prinzip beliebig gewählt werden. Die folgende Reihenfolge beinhaltet die wenigsten Arbeitsschritte:

1. Datenbank anlegen
2. Gruppen anlegen
3. Benutzer anlegen und Gruppen als Mitglied zuordnen
4. Mandanten anlegen und Gruppen sowie Benutzer zuweisen

2.11.3. Datenbanken anlegen

Zuerst muss eine Datenbank angelegt werden. Verwenden Sie für den Datenbankzugriff den vorhin angelegten Benutzer (in unseren Beispielen ist dies ‘`kivitendo`’).

2.11.4. Gruppen anlegen

Eine Gruppe wird in der Gruppenverwaltung angelegt. Ihr muss ein Name gegeben werden, eine Beschreibung ist hingegen optional. Nach dem Anlegen können Sie die verschiedenen Bereiche wählen, auf die Mitglieder dieser Gruppe Zugriff haben sollen.

Benutzergruppen werden zwar in der Authentifizierungsdatenbank gespeichert, gelten aber nicht automatisch für alle Mandanten. Der Administrator legt vielmehr fest, für welche Mandanten eine Gruppe gültig ist. Dies kann entweder beim Bearbeiten der Gruppe geschehen ("diese Gruppe ist gültig für Mandanten X, Y und Z"), oder aber wenn man einen Mandanten bearbeitet ("für diesen Mandanten sind die Gruppen A, B und C gültig").

Wurden bereits Benutzer angelegt, so können hier die Mitglieder dieser Gruppe festgelegt werden ("in dieser Gruppe sind die Benutzer X, Y und Z Mitglieder"). Dies kann auch nachträglich beim Bearbeiten eines Benutzers geschehen ("dieser Benutzer ist Mitglied in den Gruppen A, B und C").

2.11.5. Benutzer anlegen

Beim Anlegen von Benutzern werden für viele Parameter Standardeinstellungen vorgenommen, die den Gepflogenheiten des deutschen Raumes entsprechen.

Zwingend anzugeben ist der Loginname. Wenn die Passwortauthentifizierung über die Datenbank eingestellt ist, so kann hier auch das Benutzerpasswort gesetzt bzw. geändert werden. Ist hingegen die LDAP-Authentifizierung aktiv, so ist das Passwort-Feld deaktiviert.

Hat man bereits Mandanten und Gruppen angelegt, so kann hier auch konfiguriert werden, auf welche Mandanten der Benutzer Zugriff hat bzw. in welchen Gruppen er Mitglied ist. Beide Zuweisungen können sowohl beim Benutzer vorgenommen werden ("dieser Benutzer hat Zugriff auf Mandanten X, Y, Z" bzw. "dieser Benutzer ist Mitglied in Gruppen X, Y und Z") als auch beim Mandanten ("auf diesen Mandanten haben Benutzer A, B und C Zugriff") bzw. bei der Gruppe ("in dieser Gruppe sind Benutzer A, B und C Mitglieder").

2.11.6. Mandanten anlegen

Ein Mandant besteht aus Administrationssicht primär aus einem eindeutigen Namen. Weiterhin wird hier hinterlegt, welche Datenbank als Mandantendatenbank benutzt wird. Hier müssen die Zugriffsdaten einer der eben angelegten Datenbanken eingetragen werden.

Hat man bereits Benutzer und Gruppen angelegt, so kann hier auch konfiguriert werden, welche Benutzer Zugriff auf den Mandanten haben bzw. welche Gruppen für den Mandanten gültig sind. Beide Zuweisungen können sowohl beim Mandanten vorgenommen werden ("auf diesen Mandanten haben Benutzer X, Y und Z Zugriff" bzw. "für diesen Mandanten sind die Gruppen X, Y und Z gültig") als auch beim Benutzer ("dieser Benutzer hat Zugriff auf Mandanten A, B und C") bzw. bei der Gruppe ("diese Gruppe ist für Mandanten A, B und C gültig").

2.12. Drucker- und Systemverwaltung

Im Administrationsmenü gibt es ferner noch die beiden Menüpunkte Druckeradministration und System.

2.12.1. Druckeradministration

Unter dem Menüpunkt Druckeradministration lassen sich beliebig viele "Druckbefehle" im System verwalten. Diese Befehle werden mandantenweise zugeordnet. Unter Druckerbeschreibung wird der Namen des Druckbefehls festgelegt, der dann in der Druckerauswahl des Belegs angezeigt wird.

Unter Druckbefehl definiert man den eigentlichen Druckbefehl, der direkt auf dem Webserver ausgeführt wird, bspw. 'lpr -P meinDrucker' oder ein kompletter Pfad zu einem Skript (/usr/local/src/kivitendo/scripts/pdf_druck_in_verzeichnis.sh). Wird ferner noch ein optionales Vorlagenkürzel verwendet, wird dieses Kürzel bei der Auswahl der Druckvorlagendatei mit einem Unterstrich ergänzt, ist bspw. das Kürzel 'epson_drucker' definiert, so wird beim Ausdruck eines Angebots folgende Vorlage geparst: sales_quotation_epson_drucker.tex.

2.12.2. System sperren / entsperren

Unter dem Menüpunkt System gibt es den Eintrag 'Installation sperren/entsperren'. Setzt man diese Sperre so ist der Zugang zu der gesamten kivitendo Installation gesperrt.

Falls die Sperre gesetzt ist, erscheint anstelle der Anmeldemaske die Information: 'kivitendo ist momentan zwecks Wartungsarbeiten nicht zugänglich.'.

Wichtig zu erwähnen ist hierbei noch, dass sich kivitendo automatisch 'sperrt', falls es bei einem Versionsupdate zu einem Datenbankfehler kam. Somit kann hier nicht aus Versehen mit einem inkonsistenten Datenbestand weitergearbeitet werden.

2.13. E-Mail

kivitendo kann sowohl E-Mails direkt aus dem Programm heraus E-Mails versenden, als auch in das Programm importiern. Damit dies funktioniert, müssen die E-Mail-Parameter korrekt konfiguriert sein. Dazu mehr in den folgenden Abschnitten.

2.13.1. E-Mail-Versand aus kivitendo heraus

Das Versenden von E-Mails kann zum Verschicken von Belegen genutzt werden, z.B. um ein Angebot direkt an einen Kunden zu verschicken. Damit dies funktioniert, muss eingestellt werden, über welchen Server die E-Mails verschickt werden sollen. kivitendo unterstützt dabei zwei Mechanismen: Versand über einen lokalen E-Mail-Server (z.B. mit Postfix™ oder Exim™, was auch die standardmäßig aktive Methode ist) sowie Versand über einen SMTP-Server (z.B. der des eigenen Internet-Providers).

Welche Methode und welcher Server verwendet werden, wird über die Konfigurationsdatei `config/kivitendo.conf` festgelegt. Dort befinden sich alle Einstellungen zu diesem Thema im Abschnitt '`[mail_delivery]`'.

2.13.1.1. Versand über lokalen E-Mail-Server

Diese Methode bietet sich an, wenn auf dem Server, auf dem kivitendo läuft, bereits ein funktionsfähiger E-Mail-Server wie z.B. Postfix™, Exim™ oder Sendmail™ läuft.

Um diese Methode auszuwählen, muss der Konfigurationsparameter '`method = sendmail`' gesetzt sein. Dies ist gleichzeitig der Standardwert, falls er nicht verändert wird.

Um zu kontrollieren, wie das Programm zum Einliefern gestartet wird, dient der Parameter '`sendmail = ...`'. Der Standardwert verweist auf das Programm `/usr/bin/sendmail`, das bei allen oben genannten E-Mail-Serverprodukten für diesen Zweck funktionieren sollte.

Die Konfiguration des E-Mail-Servers selber würde den Rahmen dieses sprengen. Hierfür sei auf die Dokumentation des E-Mail-Servers verwiesen.

2.13.1.2. Versand über einen SMTP-Server

Diese Methode bietet sich an, wenn kein lokaler E-Mail-Server vorhanden ist oder einer vorhanden, dieser aber nicht konfiguriert ist.

Um diese Methode auszuwählen, muss der Konfigurationsparameter '`method = smtp`' gesetzt sein. Die folgenden Parameter dienen dabei der weiteren Konfiguration:

`hostname`
: Name oder IP-Adresse des SMTP-Servers. Standardwert: '`localhost`'

`port`
: Portnummer. Der Standardwert hängt von der verwendeten Verschlüsselungsmethode ab. Gilt '`security = none`' oder '`security = tls`', so ist 25 die Standardportnummer. Für '`security = ssl`' ist 465 die Portnummer. Muss normalerweise nicht geändert werden.

`security`
: Wahl der zu verwendenden Verschlüsselung der Verbindung mit dem Server. Standardwert ist '`none`', wodurch keine Verschlüsselung verwendet wird. Mit '`tls`' wird TLS-Verschlüsselung eingeschaltet, und mit '`ssl`' wird Verschlüsselung via SSL eingeschaltet. Achtung: Für '`tls`' und '`ssl`' werden zusätzliche Perl-Module benötigt (siehe unten).

`login` und `password`
: Falls der E-Mail-Server eine Authentifizierung verlangt, so können mit diesen zwei Parametern der Benutzername und das Passwort angegeben werden. Wird Authentifizierung verwendet, so sollte aus Sicherheitsgründen auch eine Form von Verschlüsselung aktiviert werden.

2.13.2. Versendete E-Mails über IMAP exportieren

Es können versendete E-Mails über IMAP exportiert werden. Dazu muss der entsprechende Server, auf dem die E-Mails abgelegt werden sollen, konfiguriert werden. Dies geschieht in der Konfigurationsdatei `config/kivitendo.conf` im Abschnitt '`[sent_emails_in_imap]`'. Es können auch Server für spezifische E-Mail-Adressen konfiguriert werden, indem der Abschnitt kopiert wird und mit dem Namen '`[sent_emails_in_imap/email/EMAILADDRESS]`', wobei '`EMAILADDRESS`' die entsprechenden E-Mail-Adresse ist, versehen wird.

Die Reihenfolge für die Auswahl der Konfiguration ist wie folgt:

1. Falls es für die Absender-Adresse eine Konfiguration gibt nimm diese.
2. Falls es es für die benutzerbezogene E-Mail-Adresse eine Konfiguration gibt dann nimm diese.
3. Falls es eine generelle Konfiguration gibt dann nimm diese.

Die folgenden Parameter dienen der Konfiguration:

`enabled`
: Falls '`enabled = 0`' gesetzt ist, wird der Export von E-Mails deaktiviert. Dies ist der Standardwert. Falls '`enabled = 1`' gesetzt ist, wird ist der Export von E-Mails aktiviert.

`hostname`
: Name oder IP-Adresse des IMAP-Servers. Standardwert: '`localhost`'

`port`
: Portnummer. Der Standardwert hängt von der verwendeten Verschlüsselungsmethode ab. Gilt '`ssl = 1`', so ist 993 die Standardportnummer. Für '`ssl = 0`' ist 143 die Portnummer. Muss normalerweise nicht geändert werden.

`ssl`
: Wahl ob eine Verschlüsselung bei der Verbindung mit dem Server verwendet wird. Standardwert ist '`1`', wodurch eine SSL-Verschlüsselung verwendet wird. Mit '`0`' wird keine Verschlüsselung genutzt.

`username` und `password`
: Falls der IMAP-Server eine Authentifizierung verlangt, so können mit diesen zwei Parametern der Nutzername und das Passwort angegeben werden. Wird Authentifizierung verwendet, so sollte aus Sicherheitsgründen auch Verschlüsselung aktiviert werden.

`folder`
: Wahl des Ordners, in den kivitendo gesendete E-Mails gespeichert. Standardwert ist '`Sent/Kivitendo`'. Ordnerhierarchien können mit einem Slash ('`/`') getrennt werden. Der gewählte Ordner muss existieren.

Bei einem Fehler bricht die Funktion ab und der Anwender bekommt die Fehlermeldung, dass die E-Mail prinzipiell nicht verschickt werden konnte. An dieser Stelle muss der kivitendo-Admin etwas genauer prüfen, ob der Mail-Versand und/oder die Synchronisation den Fehler verursacht hat. Ein Blick ins Email-Journal kann helfen den Fehler einzuschränken.

2.13.3. E-Mails in kivitendo importieren

Um E-Mails in kivitendo zu importieren, muss der entsprechende Server, von dem die E-Mails abgeholt werden sollen, konfiguriert werden. Dies geschieht in der Konfigurationsdatei `config/kivitendo.conf` im Abschnitt '`[imap_client]`'.

Die folgenden Parameter dienen der Konfiguration:

`enabled`
: Falls '`enabled = 0`' gesetzt ist, wird der Import von E-Mails deaktiviert. Dies ist der Standardwert. Falls '`enabled = 1`' gesetzt ist, wird ist der Import von E-Mails aktiviert.

`hostname`
: Name oder IP-Adresse des IMAP-Servers. Standardwert: '`localhost`'

`port`
: Portnummer. Der Standardwert hängt von der verwendeten Verschlüsselungsmethode ab. Gilt '`ssl = 1`', so ist 993 die Standardportnummer. Für '`ssl = 0`' ist 143 die Portnummer. Muss normalerweise nicht geändert werden.

`ssl`
: Wahl, ob eine Verschlüsselung bei der Verbindung mit dem Server verwendendet wird. Standardwert ist '`1`', wodurch eine SSL-Verschlüsselung verwendet wird. Mit '`0`' wird keine Verschlüsselung genutzt.

`username` und `password`
: Falls der IMAP-Server eine Authentifizierung verlangt, so können mit diesen zwei Parametern der Nutzername und das Passwort angegeben werden. Wird Authentifizierung verwendet, so sollte aus Sicherheitsgründen auch Verschlüsselung aktiviert werden.

`base_folder`
: Wahl des Ordners, den kivitendo als Basis für das Erstellen von Unterordnern und das Abholen von E-Mails verwendet. Standardwert ist '`INBOX`'. Ordnerhierarchien können mit einem Slash ('`/`') getrennt werden. Beispiel: '`INBOX/kivitendo`'. In diesem Ordner dürfen keine Unterordner existieren und keine manuell angelegt werden.

Wenn eingeschaltet, wird beim Anlegen von Verkaufsangeboten und -aufträgen ein Unterordner im E-Mail-Client für den entsprechenden Beleg angelegt. E-Mails, in diesen Ordner werden automatisch beim Öffnen des Belegs in kivitendo importiert. Um die E-Mails übersichtlich zu halten, kann der Hintergrund-Job '`CleanUpEmailSubfolders`' genutzt werden. Dieser importiert alle E-Mails und löscht alle Unterordnern, die nicht zu einem offenen Beleg gehören. Importierte E-Mails werden nicht gelöscht; erst beim Löschen des Unterordners werden sie gelöscht.

Es können auch E-Mails in kivitendo importiert werden, die nicht zu einem Beleg gehören. Dafür kann der Hintergrund-Job '`SyncEmailFolder`' verwendet werden. Dieser synchronisiert standartmäßig den Ordner der in '`base_folder`' angegeben ist. Dies kann mit dem Job-Parameter '`folder`' geändert. Die Importierten E-Mails werden im E-Mail-Journal gespeichert.

Beim Umzug des E-Mail-Servers kann kivitendo schon importierte E-Mails nicht mehr erkennen. Dies führt dazu, dass alle E-Mails erneut importiert werden.

2.14. Drucken mit kivitendo

Das Drucksystem von kivitendo benutzt von Haus aus LaTeX-Vorlagen. Um drucken zu können, braucht der Server ein geeignetes LaTeX System. Am einfachsten ist dazu eine `texlive` Installation. Unter debianoiden Betriebssystemen installiert man die Pakete mit:

```
apt install texlive-base-bin texlive-latex-recommended texlive-fonts-recommended \
  texlive-latex-extra texlive-lang-german ghostscript latexmk
```

Für Fedora benötigen Sie die folgenden Pakete:

```
dnf install texlive-collection-latex texlive-collection-latexextra \
  texlive-collection-latexrecommended texlive-collection-langgerman \
  texlive-collection-langenglish
```

Für openSUSE benötigen Sie die folgenden Pakete:

```
zypper install texlive-collection-latex texlive-collection-latexextra \
  texlive-collection-latexrecommended texlive-collection-langgerman \
  texlive-collection-langenglish
```

Anmerkung

kivitendo erwartet eine aktuelle TeX Live Umgebung, um PDF/A zu erzeugen. Aktuelle Distributionen von 2020 erfüllen diese. Überprüfbar ist dies mit dem Aufruf des installation_check.pl mit Parameter -l:

```
scripts/installations_check.pl -l
```

kivitendo bringt drei alternative Vorlagensätze mit:

- RB
- marei
- rev-odt

Der ehemalige Druckvorlagensatz "f-tex" wurde mit der Version 3.5.6 entfernt, da er nicht mehr gepflegt wird.

2.14.1. Vorlagenverzeichnis anlegen

Es lässt sich ein initialer Vorlagensatz erstellen. Die LaTeX-System-Abhängigkeiten hierfür kann man prüfen mit:

```
./scripts/installation_check.pl -lv
```

Der Angemeldete Benutzer muss in einer Gruppe sein, die über das Recht "Konfiguration -> Mandantenverwaltung" verfügt. Siehe auch Abschnitt 2.11.4, „Gruppen anlegen“ [28].

Im Userbereich lässt sich unter: "System -> Mandantenverwaltung -> Verschiedenes" die Option "Neue Druckvorlagen aus Vorlagensatz erstellen" auswählen.

1. `Vorlagen auswählen`: Wählen Sie hier den Vorlagensatz aus, der kopiert werden soll (`RB`, `marei` oder `odt-rev`.)
2. `Neuer Name`: Der Verzeichnisname für den neuen Vorlagensatz. Dieser kann im Rahmen der üblichen Bedingungen für Verzeichnisnamen frei gewählt werden.

Nach dem Speichern wird das Vorlagenverzeichnis angelegt und ist für den aktuellen Mandanten ausgewählt. Der gleiche Vorlagensatz kann, wenn er mal angelegt ist, bei mehreren Mandanten verwendet werden. Eventuell müssen Anpassungen (Logo, Erscheinungsbild, etc) noch vorgenommen werden. Den Ordner findet man im Dateisystem unter `./templates/[Neuer Name]`

2.14.2. Der Druckvorlagensatz marei

Anmerkung

Die aktuelle Dokumentation inkl. dem Foliensatz vom kivi-Treffen im August 2023 befindet sich hier: https://peitex.de/materialien/2023-08-04_kivitendo/[13]

2.14.2.1. Quickstart – Wo kann was angepasst werden?

Anmerkung

In keinem Fall sollten Dateien mit der Endung `*.cls` oder `*.sty` geändert werden. Durch Änderungen an diesen Dateien verhindert man Updates auf neuer Versionen. Zwar kopiert kivitendo die Datei und bearbeitet diese beim Update nicht. Allerdings sind sämtliche Änderungen über die Konfigurationsdateien möglich und erhöhen die Wartbarkeit.

- Pfad zu Angaben über Mandant*innen (default: firma)
- Logo/Briefpapier, falls für alle Mandant*innen in gleicher Struktur. Sonst in der `ident.tex`.
- Layout der Kopf/Fußzeile
- innerhalb dieser Datei werden auch die folgenden Dateien geladen:

[13] https://peitex.de/materialien/2023-08-04_kivitendo/

- `firma/ident.tex` Mandant*innenspezifische Konfiguration, Adressdaten
- `firma/$währungskürzel_account.tex`
- `insettings.tex` :
- Sprache/Übersetzungen.\\Es muss mindestens eine Sprache angelegt werden!

```
\item[deutsch.tex] Textschnipsel für Deutsch\\
                      Dafür eine Sprache mit Vorlagenkürzel DE anlegen
\item[english.tex] Textschnipsel für Englisch\\
                      Dafür eine Sprache mit Vorlagenkürzel EN anlegen
```

Alle dokumententypspezifischen Einstellungen müssen in der jeweiligen Template-Datei modifiziert werden.

2.14.2.2. Aufbau

Die Grundstruktur besteht je Dokumententyp aus einer Basisdatei und verschiedenen Setup-Dateien.

Die Basis wurde so überarbeitet, dass Dokumente nun generell auf der Dokumentenklasse `scrartcl.cls` basieren und das Paket `kiviletter.sty` benutzen.

Mandant*innenspezifische Konfiguration findet sich in der Datei `insettings.tex` und dem Ordner eines spezifischen Mandant*innen (default=*firma/*).

2.14.2.3. Mandant*innen / Firma

Um gleiche Vorlagen für verschiedene Firmen verwenden zu können, wird je nach dem Wert der Kivitendo-Variablen \kivivar{kivicompany} ein Firmenverzeichnis ausgewählt (siehe `insettings.tex`), in dem Briefkopf, Identitäten und Währungs-/Kontoeinstellungen hinterlegt sind. \kivivar{kivicompany} enthält den Namen des verwendeten Mandant*innendaten. Ist kein Firmenname eingetragen, so wird das generische Unterverzeichnis *firma* verwendet.

2.14.2.4. Identitäten

In jedem Firmen-Unterverzeichnis soll eine Datei `ident.tex` vorhanden sein, die mit |\newcommand| Werte für |\telefon|, |\fax|, |\firma|, |\strasse|, |\ort|, |\ustid|, |\email| und |\homepage| definiert.

2.14.2.5. Währungen/Konten

Für jede Währung (siehe `insettings.tex`) soll eine Datei vorhanden sein, die das Währungssymbol (|\currency|) und folgende Angaben für ein Konto in dieser Währung enthält |\kontonummer|, |\bank|, |\bankleitzahl|, |\bic| und |\iban|. So kann in den Dokumenten je nach Währung ein anderes Konto angegeben werden. Nach demselben Schema können auch weitere, alternative Bankverbindungen angelegt werden, die dann in `insettings.tex` als Variable in der Fußzeile eingefügt werden.

Als Fallback (falls kivitendo keine Währung an das Druckvorlagen-System übergibt) ist Euro eingestellt. Dies lässt sich in der `insettings.tex` über das optionale Argument von |\setupCurrencyConfig| anpassen, z.B.

```
\setupCurrencyConfig[chf]{\identpath}{\lxcurrency}
```

für Schweizer Franken als Standardwährung.

2.14.2.6. Briefbogen/Logos

Eine Hintergrundgrafik oder ein Logo kann in Abhängigkeit vom Medium (z.B. nur beim Verschicken mit E-Mail) eingebunden werden.

Desweiteren sind (auskommentierte) Beispiele enthalten für eine Grafik als Briefkopf, nur ein Logo, oder ein komplettes A4-PDF als Briefpapier.

Absolute Positionierung innerhalb des Brief-Layouts ist über die entsprechende Dokumentation des scrlayer-Paketes möglich. Da die Voreinstellungen bereits einige Sonderfälle automatisch berücksichtigen ist mit den Anpassungen Vorsicht geboten. Sämtliche Einstellungen sollten jedoch außerhalb der *.sty-Dateien vorgenommen werden. Anpassungen der `insettings.tex` betreffen hierbei alle Mandant*innen. Spezifischere Einstellungen sind über die zugehörige Konfigurationsdatei (`ident.tex`) möglich. In diesem Fall kann zum Ende der insettings eine weitere Konfigurationsdatei über die Verwendung von |\identpath| geladen werden. Ein Beispiel ist in der `insettings.tex` enthalten.

2.14.2.7. Fußzeile

Die Tabelle im Fuß verwendet die Angaben aus `firma/ident.tex` und *firma/`\_account.tex`. Ihre Struktur wird in der `insettings.tex` definiert. Sie kann anschließend auch Mandant*innenspezifisch überschrieben werden.

2.14.2.8. Seitenstil/Basislayout

Das Seitenlayout wird über \pck{scrlayer-scrpage} bestimmt. Die ausführliche Dokumentation findet sich in \cite{scrguide}. Es existieren in der Datei `insettings.tex` einige Hinweise zu den Anpassungen. Die Basiskonfiguration ist ebenfalls dort eingetragen.

Neben den in Abschnitt \ref{sec:options} beschriebenen Optionen zum Abschalten der Fußzeile kann der Inhalt der Fußzeile über die \pck{scrlayer-scrpage} Makros, wie

```
\cfoot[|\meta{Inhalt auf der ersten Briefseite}|]{|\meta{Inhalt auf folgenden Briefs
```

geändert werden.

Die Kopfzeile unterscheidet sich von Dokumententyp zu Dokumententyp leicht, da diese über Datenbankvariablen befüllt wird. Hierfür wird das Makro |\ourhead| in der `insettings.tex` definiert.

```
\DescribeMacro{\ourhead}\marg{Bezeichner}\marg{Eintrag}\marg{Titel}\marg{Nummer}\mar
```

Diese Definition kann ebenfalls über die `insettings.tex` angepasst oder auch nachträglich überschrieben werden:

```
\newcommand{\ourhead}[5] {%
  \chead{%
    \makebox[\textwidth]{%
      \Ifstr{#1}{}{}{#1: #2 \hspace{0.7cm}}%
      #3%
      \Ifstr{#4}{}{}{~\nr: #4}%
      \Ifstr{#5}{}{}{\vom ~ #5}%
      \hspace{0.7cm} - \seite ~ \thepage/\letterlastpage  ~-%
    }%
  }%
}
```

In der Standard-Einstellung sieht eine Kopfzeile mit obigen Aufruf dann folgendermaßen aus:

```
\newcommand{\ourhead}[5] {%
    \makebox[\textwidth]{%
      \Ifstr{#1}{}{}{#1: #2 \hspace{0.7cm}}%
      #3%
      \Ifstr{#4}{}{}{~\nr: #4}%
      \Ifstr{#5}{}{}{\vom ~ #5}%
```

```
      \hspace{0.7cm} - \seite ~ \thepage/\letterlastpage  ~-%
    }%
}

\begingroup
\def\letterlastpage{50}

\ourhead{arg1}{arg2}{arg3}{arg4}{arg5}

\smallskip
Erzeugt mit dem Aufruf
\ourhead{arg1}{arg2}{arg3}{arg4}{arg5}
\endgroup
```

2.14.2.9. Absenderergänzung

Die Absenderergänzung wird über die Variable |location| in der |kiviletter.sty| folgendermaßen belegt:

```
\setkomavar{location}{
  \Ifkomavarempty{transaction}{}{{
        \usekomafont{transaction}
        \usekomavar{transaction}
      }
  }
  \par
  \medskip
  \parbox{\useplength{locwidth}}{
    \locationentry{date}
    \locationentry{myref}
    \locationentry{customer}
    \locationentry{yourref}
    \locationentry{delivery}
    \locationentry{quote}
    \locationentry{orderID}
    \locationentry{projectID}
    \locationentry{taxpoint}
    \locationentry[\ansprechpartner]{fromname}
    \locationentry{fromphone}
    \locationentry*{fromemail}
  }
}
```

Um die Reihenfolge der Variablenausgabe zu verändern, kann diese Definition als Basis in die `insettings.tex` oder `ident.tex` (Falls nur für eine Firma) kopiert und dort entsprechend modifiziert werden.

Das Vorgehen geht für alle vorbelegten Variablen analog.

2.14.2.10. Allgemeine TeXnische Hinweise

2.14.2.10.1. Änderung der Basisschriftart

\LaTeX{} kann grundsätzlich beliebige Schriftarten verwenden. Hierfür sollte allerdings immer darauf geachtet werden, dass die Lizenz der Schriftart das einbetten von Glyphen erlaubt. Dies liegt in der Verantwortung der Anwender*innen. Darüber hinaus ist wichtig, welches Kompilierungsprogramm verwendet werden muss. Um TrueType oder OpenType Schriftarten zu nutzen sollte |lualatex| verwendet werden. Bei Type1 Schriftarten, die speziell für \LaTeX{} installiert wurden, ist pdfLaTeX möglich. Da man heutzutage nur noch moderne Schriftformate Verwenden sollte, wird hier lediglich die Variante für |lualatex| aufgelistet.

Die Konfiguration läuft hierbei über das \pck{fontspec} Paket (Doku siehe \cite{fontspec}). Dann hängt es davon ab, ob die Basisschriftart eine Serifenschriftart ist oder nicht. In jedem Fall wird die Änderung entweder in der `insettings.tex`, sofern sie für alle Mandant*innen gelten soll oder in der Mantant*innenspezifischen Konfigurationsdatei gsesetzt.

2.14.2.10.1.1. Änderung, falls es ein Schriftpaket gibt

Wenn möglich sollte die Schriftart über ein entsprechendes Konfigurationspaket gesetzt werden. Ob ein solches existiert kann man sehr leicht über eine Suche nach dem Namen unter \url{ctan.org} herausfinden.

2.14.2.10.1.2. Änderung der Basisschriftart auf eine Schriftart mit Serifen

```
\setmainfont{|\meta{Name der Schriftart, z.B. SourceSerifPro}|}
```

2.14.2.10.1.3. Änderung der Basisschriftart auf eine Schriftart ohne Serifen

```
\setsansfont{|\meta{Name der Schriftart, z.B. SourceSansPro}|}
\renewcommand*{\familydefault}{\sfdefault}
```

2.14.2.10.2. Unterscheidungen durch String-Vergleich

```
\Ifstr{\lxmedia}{printer}{Falls gedruckt werden soll} {sonst}
```

2.14.3. Der Druckvorlagensatz mersiha

Hierbei handelt es sich um einen auf Industriekunden zugeschnitten LaTeX Druckvorlagensatz auf Basis des Druckvorlagensatzes marei.

- Die Felder in der Infobox oben rechts auf den Belegen lassen sich flexibel anpassen
- Belegtitel (Überschriften) sind linksbündig und kleiner als im LaTeX KOMA-script Standard
- Als optional markierten Positionen sind auf den Belegen als solche gekennzeichnet

Die Dokumentenklasse kiviletter.sty wird nicht modifiziert, um die Wartung zu erleichtern.

2.14.3.1. Struktur der Belegtypen

Der Druckvorlagensatz ist so strukturiert, dass für jeden Belegtyp in kivitendo eine zugehörige LaTeX-Quelltextdatei existiert, welche beim Drucken automatisch in ein PDF umgewandelt wird. Grundsätzlich werden Doppelungen im LaTeX-Code vermieden, um die Wartung zu vereinfachen. Belegtypen, die sich nur minimal voneinander unterscheiden, verweisen auf dieselbe LaTeX-Quelltextdatei, welche dann Fallunterscheidungen, z.B. für die Überschrift, enthält. Eine Übersicht aller Belegtypen und zugehöriger LaTeX-Quelltextdateien liefert folgende Tabelle. Dateinamen innerhalb derselben Tabellenzelle verweisen mittels symbolischer Links auf dieselbe Datei.

Tabelle 2.1. Dateinamen der Belegtypen

Einkauf		**Verkauf**	
Belegtyp	**LaTeX Vorlage**	**Belegtyp**	**LaTeX Vorlage**
		Pflichtenheft	requirement_spec.tex
Preisanfrage	request_quotation.tex	Angebot	sales_quotation.tex
Angebotseingang	purchase_quotation_intake.tex	Auftragseingang	sales_order_intake.tex
Lieferantenauftrag	purchase_order.tex	Auftragsbestätigung	sales_order.tex*
Beistelllieferschein	supplier_delivery_order.tex	Lieferschein	sales_delivery_order.tex

Einkauf		Verkauf	
Belegtyp	**LaTeX Vorlage**	**Belegtyp**	**LaTeX Vorlage**
Retourenlieferschein Einkaufslieferschein	rma_delivery_order.tex* purchase_delivery_order.tex*		
Einkaufsreklamation	purchase_reclamation.tex	Verkaufsreklamation	sales_reclamation.tex
		Rechnung Anzahlungsrechnung Schlussrechnung Rechnungskopie Stornorechnung	invoice.tex invoice_for_advance_payment.tex* final_invoice.tex* invoice_copy.tex* storno_invoice.tex*
		Proformarechnung	proforma.tex
		Gutschrift	credit_note.tex
		Mahnung	zahlungserinnerung.tex
		Rechnung über Mahngebühren	zahlungserinnerung_invoice.tex
		Sammelrechnung (FiBu)	statement.tex

*: symbolischer Link

Beim Drucken eines Beleges lässt sich im Druckdialog abhängig vom Belegtyp eine von mehreren Druckvarianten auswählen.

Tabelle 2.2. Druckvarianten der Belegtypen (sofern Varianten vorhanden sind)

Belegtyp	Druckvariante	LaTeX-Vorlage
Lieferschein	Sammelliste	pick_list.tex
Lieferantenauftrag	Lagerliste	bin_list.tex
Angebot	Kostenvoranschlag	sales_estimate.tex
Auftragsbestätigung, Rechnung	Proforma	proforma.tex

Desweiteren werden das Logo oder ein Briefbogen, die Daten im Identitätsbereich in der Fußzeile, einschließlich Kontodaten für verschiede Währungen, und Übersetzungen aller Begriffe zentral eingestellt und sind dann für alle Belege wirksam. Dies passiert in den Dateien:

```
deutsch.tex
englisch.tex
firma/chf_account.tex
firma/euro_account.tex
firma/usd_account.tex
firma/ident.tex
insettings.tex
```

2.14.3.2. Infobox

Die Infobox oben rechts auf den Belegen (auch genannt Absenderergänzung) wird über LaTeX KOMA-script Variablen befüllt. Dies ist flexibler als im Druckvorlagensatz marei gestaltet. In der Datei `insettings.tex` sind die KOMA-script Variablen `info0` bis `info9` definiert. Diese erscheinen jedoch nur dann in der Infobox, wenn sie auf einen Wert ungleich eines leeren Strings gesetzt werden. Das erlaubt es, in den Belegen ohne weitere Fallunterscheidungen die Variablen zu setzen. Zum Verständnis hier ein Beispiel aus dem Belegtyp Rechnung, also `invoice.tex`:

```
% \setkomavar{info0}[\vorgangsbez]       {<%transaction_description%>}
\setkomavar{info1}[\rechnungsdatum]    {<%invdate%>}
\setkomavar{info2}[\kundennummer]      {<%customernumber%>}
\setkomavar{info3}[\ihreBestellnummer] {<%cusordnumber%>}
\setkomavar{info4}[\auftragsnummer]    {<%ordnumber%>}
```

```
\setkomavar{info5}[\angebot{}~\nr]     {<%quonumber%>}
\setkomavar{info6}[\lieferschein{}~\nr]{<%donumber%>}
\setkomavar{info7}[\leistungsdatum]    {<%tax_point%>}
\setkomavar{info8}[\ansprechpartner]   {<%salesman_name%>}
\setkomavar{info9}[\textTelefon]       {<%salesman_tel%>}
\setkomavar{info10}[\textEmail]        {<%salesman_email%>}
```

Dieser Kunde möchte die Vorgangsbezeichnung nicht mit ausgeben, weshalb die erste Zeile auskommentiert ist. Weiterhin ist exemplarisch der Verkäufer anstelle des Bearbeiters der Ansprechpartner. Daher wird mit `<%salesman_name%>` anstelle von `<%employee_name%>` gearbeitet. Telefon und E-Mail verhalten sich analog dazu.

Weil `info0` bis `info11` in aufsteigender Reihenfolge ausgegeben werden, lässt sich über die Ziffern 0 bis 11 die Reihenfolge der Felder in der Infobox steuern. Bei vielen Industriekunden benötigen bestimmte Belegtypen abweichende Informationen oder andere Übersetzungstexte in der Infobox; all dies lässt sich durch Anpassung obigen LaTeX-Codes erreichen.

2.14.3.3. Tipps für Kundenanpassungen

- Logo mit absoluter Positionierung als Briefbogen einfügen: In `insettings.tex` einfügen:

  ```
  \DeclareNewLayer[page,background,hoffset=125mm,voffset=12mm,
    contents={\includegraphics[width=75mm]{logo-mit-text}}
  ]{background.first}
  \DeclareNewLayer[page,background,hoffset=20mm,voffset=4mm,
    contents={\includegraphics[scale=0.2]{logo-klein}}
  ]{background}
  \AddLayersToPageStyle{kivitendo.letter.first}{background.first}
  \AddLayersToPageStyle{kivitendo.letter}{background}
  ```

  Das Logo muss in einem für LaTeX geeigneten Format vorliegen, bspw. PDF, PNG oder JPEG: `logo-mit-text.pdf` o.Ä.

- Die alternierenden Farben der Zeilen in der Positionstabelle abschalten: In `insettings.tex` einfügen:

  ```
  \SetupPricingTabular{
    rowcolor-odd = white,
  }
  ```

- Im Druckvorlagensatz mersiha bereits enthalten ist: Die standardmäßig aktive Ausgabe des Währungszeichens in der letzten Spalte der Positionstabelle abschalten: In `insettings.tex` einfügen:

  ```
  \SetupPricingTabular{
    pricetotal/colspec = K,
  }
  ```

  Dies ist bspw. nötig, um optionale Positionen richtig auszugeben und das Währungszeichen nur alle nicht-optionalen Positionen durch eine Fallunterscheidung in Template Toolkit einzufügen.

- In Lieferantenaufträgen zu jeder Position die Artikelnummer des Lieferanten ausgeben: In `purchase_order.tex` folgendes diff (mit + markierte Zeilen) einfügen:

  ```
        <%foreach number%>%
          <%runningnumber%> &%
          <%number%> &%
          \textbf{<%description%>}%
  +       <%foreach make%><%if make == name%>\ExtraDescription{<%model%>}<%end%><%end%>%
          <%if longdescription%>\ExtraDescription{<%longdescription%>}<%end longdescription
          <%if reqdate%>\ExtraDescription{\lieferdatum: <%reqdate%>}<%end reqdate%>%
  ```

  Hier wird durch eine Schleife in Template Toolkit der richtige Wert extrahiert.

2.14.4. Der Druckvorlagensatz RB

Hierbei handelt es sich um einen vollständigen LaTeX Dokumentensatz mit alternativem Design. Die odt oder html-Varianten sind nicht gepflegt.

Die konzeptionelle Idee der Vorlagen wird hier[14] auf Folie 5 bis 10 vorgestellt. Informationen zur Anpassung an die eigenen Firmendaten finden sich in der Datei Readme.tex im Vorlagenverzeichnis.

Eine kurze Übersicht der Features:

- Mehrsprachenfähig, mit Deutscher und Englischer Übersetzung
- Zentrale Konfigurationsdateien, die für alle Belege benutzt werden, z.B. für Kopf- und Fußzeilen, und Infos wie Bankdaten
- mehrere vordefinierte Varianten für Logos/Hintergrundbilder
- Berücksichtigung für Steuerzonen "EU mit USt-ID Nummer" oder "Außerhalb EU"

2.14.5. Der Druckvorlagensatz rev-odt

Hierbei handelt es sich um einen Dokumentensatz der mit odt-Vorlagen erstellt wurde. Es gibt in dem Verzeichnis eine Readme-Datei, die eventuell aktueller als die Dokumentation hier ist. Die odt-Vorlagen in diesem Verzeichnis "rev-odt" wurden von revamp-it, Zürich erstellt und werden laufend aktualisiert. Ein paar der Formulierungen in den Druckvorlagen entsprechen dem Schweizer Sprachgebrauch, z.B. "Offerte" oder "allfällig".

Hinweis zum Einsatz des Feldes "Land" bei den Stammdaten für KundInnen und LieferantInnen, sowie bei Lieferadressen: Die in diesem Vorlagensatz vorhandenen Vorlagen erwarten für "Land" das entsprechende Kürzel, das in Adressen vor die Postleitzahl gesetzt wird. Das Feld kann auch komplett leer bleiben. Wer dies anders handhaben möchte, muss die Vorlagen entsprechend anpassen.

odt-Vorlagen können mit LibreOffice oder OpenOffice editiert und den eigenen Bedürfnissen angepasst werden. Wichtig beim Editieren von if-Blöcken ist, dass immer der gesamte Block überschrieben werden muss und nicht nur Teile davon, da dies sonst oft zu einer odt-Datei führt, die vom Parser nicht korrekt gelesen werden kann.

Mahnungen können unter folgenden Einschränkungen mit den odt-Vorlagen im Vorlagensatz rev-odt erzeugt werden:

- als Druckoption steht nur 'PDF(OpenDocument/OASIS)' zur Verfügung, das heisst, die Mahnungen werden als PDF-Datei ausgegeben.
- für jede Rechnung muss eine eigene Mahnung erzeugt werden (auch wenn bei einzelnen KundInnen mehrere überfällige Rechnungen vorhanden sind).

Mehrere Mahnungen für eine Kundin / einen Kunden werden zu einer PDF-Datei zusammengefasst

Die Vorlagen zahlungserinnerung.odt sowie mahnung.odt sind für das Erstellen einer Zahlungserinnerung bzw. Mahnung selbst vorgesehen, die Vorlage mahnung_invoice.odt für das Erstellen einer Rechnung über die verrechneten Mahngebühren und Verzugszinsen.

Zur Zeit gibt es in kivitendo noch keine Möglichkeit, odt-Vorlagen bei Briefen und Pflichtenheften einzusetzen. Entsprechende Vorlagen sind deshalb nicht vorhanden.

Fehlermeldungen, Anregungen und Wünsche bitte senden an: empfang@revamp-it.ch

2.14.6. Allgemeine Hinweise zu LaTeX Vorlagen

In den allermeisten Installationen sollte das Drucken jetzt schon funktionieren. Sollte ein Fehler auftreten, wirft TeX sehr lange Fehlerbeschreibungen, der eigentliche Fehler ist immer die erste Zeile, die mit einem Ausrufezeichen anfängt. Häufig auftretende Fehler sind zum Beispiel:

[14] http://www.kivitendo-support.de/vortraege/Lx-Office%20Anwendertreffen%20LaTeX-Druckvorlagen-Teil3-finale.pdf

- ! LaTeX Error: File `eurosym.sty' not found. Die entsprechende LaTeX-Bibliothek wurde nicht gefunden. Das tritt vor allem bei Vorlagen aus der Community auf. Installieren Sie die entsprechenden Pakete.
- ! Package inputenc Error: Unicode char \u8:... set up for use with LaTeX. Dieser Fehler tritt auf, wenn sie versuchen mit einer Standardinstallation exotische utf8 Zeichen zu drucken. TeXLive unterstützt von Haus nur romanische Schriften und muss mit diversen Tricks dazu gebracht werden andere Zeichen zu akzeptieren. Adere TeX Systeme wie XeTeX schaffen hier Abhilfe.

Wird gar kein Fehler angezeigt, sondern nur der Name des Templates, heißt das normalerweise, dass das LaTeX Binary nicht gefunden wurde. Prüfen Sie den Namen in der Konfiguration (Standard: `latexmk --pdflatex`), und stellen Sie sicher, dass latexmk (oder das von Ihnen verwendete System) vom Webserver ausgeführt werden darf.

Wenn sich das Problem nicht auf Grund der Ausgabe im Webbrowser verifizieren lässt:

- editiere [kivitendo-home]/config/kivitendo.conf und ändere "keep_temp_files" auf 1

  ```
  keep_temp_files = 1;
  ```

- bei fastcgi oder mod_perl den Webserver neu Starten
- Nochmal einen Druckversuch im Webfrontend auslösen
- wechsel in das users Verzeichnis von kivitendo

  ```
  cd [kivitendo-home]/users
  ```

- LaTeX Suchpfad anpassen:

  ```
  export TEXINPUTS=".:[kivitendo-home]/templates/[aktuelles_template_verzeichniss]:"
  ```

- Finde heraus, welche Datei kivitendo beim letzten Durchlauf erstellt hat

  ```
  ls -lahtr ./1*.tex
  ```

  Es sollte die letzte Datei ganz unten sein

- für besseren Hinweis auf Fehler texdatei nochmals übersetzen

  ```
  pdflatex ./1*.tex
  ```

  in der *.tex datei nach dem Fehler suchen.

2.15. OpenDocument-Vorlagen

kivitendo unterstützt die Verwendung von Vorlagen im OpenDocument-Format, wie es LibreOffice oder OpenOffice (ab Version 2) erzeugen. kivitendo kann dabei sowohl neue OpenDocument-Dokumente als auch aus diesen direkt PDF-Dateien erzeugen. Nachfolgend ist die Verwendung mit LibreOffice beschrieben. Für die Verwendung mit OpenOffice müssen die Einstellungen allenfalls angepasst werden.

2.15.1. Grundeinstellung

Um die Unterstützung von OpenDocument-Vorlagen zu aktivieren, muss in der Datei `config/kivitendo.conf` die Variable `opendocument`, im Abschnitt `print_templates`, auf `1` stehen. Dies ist die Standardeinstellung.

2.15.2. Direkte Erzeugung von PDF-Dateien

Während die Erzeugung von reinen OpenDocument-Dateien keinerlei weitere Software benötigt, wird zur Umwandlung dieser Dateien in PDF LibreOffice oder OpenOffice benötigt.

Unter Debian kann dieses installiert werden mit:

```
apt install libreoffice
```

Der Pfad zu LibreOffice kann in der Datei `config/kivitendo.conf`, im Abschnitt `applications`, angepasst werden. Unter Debian sollte dies nicht notwendig sein.

2.15.2.1. Variante 1)

In der Standardeinstellung wird LibreOffice für jedes Dokument neu gestartet. Es ist keine weitere Konfiguration erforderlich.

2.15.2.2. Variante 2)

Die zweite Variante startet ein LibreOffice, das auch nach der Umwandlung des Dokumentes gestartet bleibt. Bei weiteren Umwandlungen wird dann diese laufende Instanz benutzt. Der Vorteil ist, dass die Zeit zur Umwandlung möglicherweise reduziert wird, weil nicht für jedes Dokument eine neue Instanz gestartet werden muss.

Dazu muss in der Konfigurationsdatei `config/kivitendo.conf`, im Abschnitt `print_templates`, `openofficeorg_daemon = 1` gesetzt werden.

Diese Methode verwendet Python und die Python uno bindings. Unter Debian werden diese mit LibreOffice mitgeliefert.

Die Optionen `python_uno` und `python_uno_path` in der Konfigurationsdatei `config/kivitendo.conf` dienen zur Anpassung der Python uno Pfade. Unter Debian sollte keine Anpassung notwendig sein.

2.15.3. Vorbereitungen

2.15.3.1. Adminbereich

Damit beim Erstellen von Rechnungen und Aufträgen neben der Standardvorlage ohne Einzahlungsschein weitere Vorlagen (z.B. mit Einzahlungsschein) auswählbar sind, muss für jedes Vorlagen-Suffix ein Drucker eingerichtet werden:

- Druckeradministration → Drucker hinzufügen
- Mandant wählen
- Druckerbeschreibung → aussagekräftiger Text: wird in der Auftrags- bzw. Rechnungsmaske als Auswahl angezeigt (z.B. mit Einzahlungsschein Bank xy)
- Druckbefehl → beliebiger Text (hat für das Erzeugen von Aufträgen oder Rechnungen als odt-Datei keine Bedeutung, darf aber nicht leer sein)
- Vorlagenkürzel → besr bzw. selbst gewähltes Vorlagensuffix (muss genau der Zeichenfolge entsprechen, die zwischen "invoice_" bzw. "sales_order_" und ".odt" steht.)
- speichern

2.15.3.2. Benutzereinstellungen

Wer den Ausdruck mit Einzahlungsschein als Standardeinstellung im Rechnungs- bzw. Auftragsformular angezeigt haben möchte, kann dies persönlich für sich bei den Benutzereinstellungen konfigurieren:

- Programm → Benutzereinstellungen → Druckoptionen
- Standardvorlagenformat → OpenDocument/OASIS
- Standardausgabekanal → Bildschirm
- Standarddrucker → gewünschte Druckerbeschreibung auswählen (z.B. mit Einzahlungsschein Bank xy)
- Anzahl Kopien → leer

- speichern

2.15.3.3. Auswahl der Druckvorlage in kivitendo beim Erzeugen einer odt-Rechnung (analog bei Auftrag)

Im Fussbereich der Rechnungsmaske muss neben Rechnung, OpenDocument/OASIS und Bildschirm die im Adminbereich erstellte Druckerbeschreibung ausgewählt werden, falls diese nicht bereits bei den Benutzereinstellungen als persönlicher Standard gewählt wurde.

2.15.4. Schweizer QR-Rechnung mit OpenDocument Vorlagen

Mit der Version 3.6.0 unterstützt Kivitendo die Erstellung von Schweizer QR-Rechnungen gemäss Swiss Payment Standards, Version 2.2[15]. Implementiert sind hierbei die Varianten:

- **QR-IBAN mit QR-Referenz**
- **IBAN ohne Referenz**

2.15.4.1. Einstellungen

2.15.4.1.1. Mandantenkonfiguration

Unter *System → Mandantenkonfiguration → Features*. Im Abschnitt *Einkauf und Verkauf*, beim Punkt *Verkaufsrechnungen mit Schweizer QR-Rechnung erzeugen*, die gewünschte Variante wählen.

2.15.4.1.2. Konfiguration der Bankkonten

Unter *System → Bankkonten* muss bei mindestens einem Bankkonto die Option *Nutzung mit Schweizer QR-Rechnung* auf **Ja** gestellt werden.

Die IBAN muss ohne Leerzeichen angegeben werden.

Tipp

Für die Variante **QR-IBAN mit QR-Referenz** muss dieses Konto unter QR-IBAN (Schweiz) eine gültige **QR-IBAN Nummer** enthalten. Diese unterscheidet sich von der regulären IBAN.

Zusätzlich muss eine gültige **Bankkonto Identifikationsnummer** angegeben werden (6-stellig).

Diese werden von der jeweiligen Bank vergeben.

Sind mehrere Konten ausgewählt wird das erste verwendet.

2.15.4.1.3. Rechnungen ohne Betrag

Für Rechnungen ohne Betrag (z.B. Spenden) kann, in der jeweiligen Rechnung, die Checkbox *QR-Rechnung ohne Betrag* aktiviert werden. Diese Checkbox erscheint nur, wenn QR-Rechnungen in der Mandantenkonfiguration aktiviert sind (variante ausgewählt).

Dies wirkt sich lediglich auf den erzeugten QR-Code aus. Die Vorlage muss separat angepasst und ausgewählt werden.

2.15.4.2. Adressdaten

Die Adressdaten zum Zahlungsempfänger werden aus der Mandantenkonfiguration entnommen. Unter *System → Mandantenkonfiguration → Verschiedenes*, Abschnitt *Firmenname und -adresse.*

[15] https://www.paymentstandards.ch/dam/downloads/ig-qr-bill-de.pdf

Die Adressdaten zum Zahlungspflichtigen stammen aus den Kundendaten der jeweiligen Rechnung.

Ist bei den Adressdaten kein Land angegeben, wird standardmässig Schweiz verwendet. Akzeptiert werden Ländername oder Ländercode, also z.B. "Schweiz" oder "CH".

Die Adressdaten können in der Vorlage mit den jeweiligen Variablen eingetragen werden. Siehe auch: Abschnitt 3.3, „Dokumentenvorlagen und verfügbare Variablen“ [57]

Der erzeugte QR-Code verwendet Adress-Typ "K" (Kombinierte Adressfelder, 2 Zeilen).

2.15.4.3. Referenznummer

Die Referenznummer wird in Kivitendo erzeugt und setzt sich wiefolgt zusammen:

- Bankkonto Identifikationsnummer (6-stellig)
- Kundennummer (6-stellig, mit führenden Nullen aufgefüllt)
- Rechnungsnummer (14-stellig, mit führenden Nullen aufgefüllt)
- Prüfziffer (1-stellig, berechnet mittels modulo 10, rekursiv)

Es sind lediglich Ziffern erlaubt. Allfällige Buchstaben und Sonderzeichen werden entfernt und fehlende Stellen werden mit führenden Nullen aufgefüllt.

2.15.4.4. Vorlage

Der Vorlagensatz "rev-odt" enthält die Vorlage `invoice_qr.odt`, welche für die Erstellung von QR-Rechnungen vorgesehen ist. Damit diese verwendet werden kann muss wie obenstehend beschrieben ein Drucker hinzugefügt werden, allerdings mit dem Vorlagenkürzel `qr` (siehe Abschnitt 2.15.3.1, „Adminbereich“ [42]). Weitere Vorlagen für die QR-Rechnung müssen im Dateinamen, bzw. Vorlagenkürzel, ebenfalls die Zeichenfolge `qr` enthalten, also z.b. `invoice_qr2.odt` etc.

Die Vorlagen können beliebig angepasst werden. Zwingend müssen diese jedoch das QR-Code Platzhalter Bild, als eingebettetes Bild, enthalten. Da dieses beim Ausdrucken/Erzeugen der Rechnung durch das neu generierte QR-Code Bild ersetzt wird. Das Bild muss den Namen `QRCodePlaceholder` tragen. In Libreoffice unter *Bild-Kontextmenü* → *Einstellungen* → *Optionen* → *Name* einstellbar. Siehe dazu auch die Beispielvorlage.

2.15.4.4.1. Zusätzliche Variablen für Vorlage

Zusätzlich zu den in der Vorlage standardmässig verfügbaren Variablen (siehe Abschnitt 3.3, „Dokumentenvorlagen und verfügbare Variablen“ [57]), werden die folgenden Variablen erzeugt:

ref_number_formatted
: Referenznummer formatiert mit Leerzeichen, z.B.: 21 00000 00003 13947 14300 09017

iban_formatted
: IBAN formatiert mit Leerzeichen

amount_formatted
: Betrag formatiert mit Tausendertrennzeichen Leerschlag, z.B.: 1 005.55

2.16. Nomenklatur

2.16.1. Datum bei Buchungen

Seit der Version 3.5 werden für Buchungen in kivitendo einheitlich folgende Bezeichnungen verwendet:

- `Erfassungsdatum` (en: `Entry Date`, code: `Gldate`)

bezeichnet das Datum, an dem die Buchung in kivitendo erfasst wurde.

- `Buchungsdatum` (en: `Booking Date`, code: `Transdate`)

  bezeichnet das buchhaltungstechnisch für eine Buchung relevante Datum

  Das `Rechnungsdatum` bei Verkaufs- und Einkaufsrechnungen entspricht dem Buchungsdatum. Das heisst, in Berichten wie dem Buchungsjournal, in denen eine Spalte `Buchungsdatum` angezeigt werden kann, erscheint hier im Fall von Rechnungen das Rechnungsdatum.

- Bezieht sich ein verbuchter Beleg auf einen Zeitpunkt, der nicht mit dem Buchungsdatum übereinstimmt, so kann dieses Datum momentan in kivitendo nur unter Bemerkungen erfasst werden.

  Möglicherweise wird für solche Fälle in einer späteren Version von kivitendo ein dritter Datumswert für Buchungen erstellt. (Beispiel: Einkaufsbeleg stammt aus einem früheren Jahr, das bereits buchhaltungstechnisch abgeschlossen wurde, und muss deshalb später verbucht werden.)

2.17. Konfiguration zur Einnahmenüberschuss-rechnung/Bilanzierung: EUR

2.17.1. Einführung

kivitendo besaß bis inklusive Version 2.6.3 einen Konfigurationsparameter namens `eur`, der sich in der Konfigurationsdatei `config/kivitendo.conf` (damals noch `config/lx_office.conf`) befand. Somit galt er für alle Mandanten, die in dieser Installation benutzt wurden.

Mit der nachfolgenden Version wurde der Parameter zum Einen in die Mandantendatenbank verschoben und dabei auch gleich in drei Einzelparameter aufgeteilt, mit denen sich das Verhalten genauer steuern lässt.

2.17.2. Konfigurationsparameter

Es gibt drei Parameter, die die Gewinnermittlungsart, Versteuerungsart und die Warenbuchungsmethode regeln:

`profit_determination`
: Dieser Parameter legt die Berechnungsmethode für die Gewinnermittlung fest. Er enthält entweder `balance` für Betriebsvermögensvergleich/Bilanzierung oder `income` für die Einnahmen-Überschuss-Rechnung.

`accounting_method`
: Dieser Parameter steuert die Buchungs- und Berechnungsmethoden für die Versteuerungsart. Er enthält entweder `accrual` für die Soll-Versteuerung oder `cash` für die Ist-Versteuerung.

`inventory_system`
: Dieser Parameter legt die Warenbuchungsmethode fest. Er enthält entweder `perpetual` für die Bestandsmethode oder `periodic` für die Aufwandsmethode.

Zum Vergleich der Funktionalität bis und nach 2.6.3: `eur` = 1 bedeutete Einnahmen-Überschuss-Rechnung, Ist-Versteuerung und Aufwandsmethode. `eur` = 0 bedeutete hingegen Bilanzierung, Soll-Versteuerung und Bestandsmethode.

Die Konfiguration `"eur"` unter `[system]` in der Konfigurationsdatei `config/kivitendo.conf` wird nun nicht mehr benötigt und kann entfernt werden. Dies muss manuell geschehen.

2.17.3. Festlegen der Parameter

Beim Anlegen eines neuen Mandanten bzw. einer neuen Datenbank in der Admininstration können diese Optionen nun unabhängig voneinander eingestellt werden.

Für die Schweiz sind folgende Einstellungen üblich:

- Sollversteuerung
- Aufwandsmethode
- Bilanzierung

Diese Einstellungen werden automatisch beim Erstellen einer neuen Datenbank vorausgewählt, wenn in `config/kivitendo.conf` unter `[system]` `default_manager = swiss` eingestellt ist.

Beim Upgrade bestehender Mandanten wird eur ausgelesen und die Variablen werden so gesetzt, daß sich an der Funktionalität nichts ändert.

Die aktuelle Konfiguration wird unter Nummernkreise und Standardkonten unter dem neuen Punkt "Einstellungen" (read-only) angezeigt. Unter System → Mandantenkonfiguration können die Einstellungen auch geändert werden. Dabei ist zu beachten, dass eine Änderung vorhandene Daten so belässt und damit evtl. die Ergebnisse verfälscht. Dies gilt vor Allem für die Warenbuchungsmethode (siehe auch Bemerkungen zur Bestandsmethode).

2.17.4. Bemerkungen zur Bestandsmethode

Die Bestandsmethode ist eigentlich eine sehr elegante Methode, funktioniert in kivitendo aber nur unter bestimmten Bedingungen: Voraussetzung ist, daß auch immer alle Einkaufsrechnungen gepflegt werden, und man beim Jahreswechsel nicht mit einer leeren Datenbank anfängt, da bei jedem Verkauf anhand der gesamten Rechnungshistorie der Einkaufswert der Ware nach dem FIFO-Prinzip aus den Einkaufsrechnungen berechnet wird.

Die Bestandsmethode kann vom Prinzip her also nur funktioneren, wenn man mit den Buchungen bei Null anfängt, und man kann auch nicht im laufenden Betrieb von der Aufwandsmethode zur Bestandsmethode wechseln.

2.17.5. Bekannte Probleme

Bei bestimmten Berichten kann man derzeit noch inviduell einstellen, ob man nach Ist- oder Sollversteuerung auswertet, und es werden im Code Variablen wie $accrual oder $cash gesetzt. Diese Codestellen wurden noch nicht angepasst, sondern nur die, wo bisher die Konfigurationsvariable `$::lx_office_conf{system}->{eur}` ausgewertet wurde.

Es fehlen Hilfetext beim Neuanlegen eines Mandanten, was die Optionen bewirken, z.B. mit zwei Standardfällen.

2.18. SKR04 19% Umstellung für innergemeinschaftlichen Erwerb

2.18.1. Einführung

Die Umsatzsteuerumstellung auf 19% für SKR04 für die Steuerschlüssel "EU ohne USt-ID Nummer" ist erst 2010 erfolgt. kivitendo beinhaltet ein Upgradeskript, das das Konto 3804 automatisch erstellt und die Steuereinstellungen korrekt einstellt. Hat der Benutzer aber schon selber das Konto 3804 angelegt, oder gab es schon Buchungen im Zeitraum nach dem 01.01.2007 auf das Konto 3803, wird das Upgradeskript vorsichtshalber nicht ausgeführt, da der Benutzer sich vielleicht schon selbst geholfen hat und mit seinen Änderungen zufrieden ist. Die korrekten Einstellungen kann man aber auch per Hand ausführen. Nachfolgend werden die entsprechenden Schritte anhand von Screenshots dargestellt.

Für den Fall, daß Buchungen mit der Steuerschlüssel "EU ohne USt.-IdNr." nach dem 01.01.2007 erfolgt sind, ist davon auszugehen, dass diese mit dem alten Umsatzsteuersatz von 16% gebucht worden sind, und diese Buchungen sollten entsprechend kontrolliert werden.

2.18.2. Konto 3804 manuell anlegen

Die folgenden Schritte sind notwendig, um das Konto manuell anzulegen und zu konfigurieren. Zuerst wird in System → Kontenübersicht → Konto erfassen das Konto angelegt.

Kontodaten bearbeiten

Grundeinstellungen

Kontonummer 3804

Beschreibung Umsatzsteuer aus EG-Erwerb 19%

Kontentyp Konto

Kontoart

Erlöskonto (I)

Buchungskonto in

☐ **Verkauf** ☐ **Einkauf** ☐ **Inventar**

In Aufklappmenü aufnehmen

Dieser Block ist nur dann gültig, wenn das Konto KEIN Buchungskonto ist, und wenn ein gültiger Steuerschlüsse

Forderungen	Verbindlichkeiten
☐ Erlöskonto	☐ Aufwand/Anlagen
☐ Zahlungseingang	☐ Zahlungsausgang
☑ Steuer	☐ Steuer

Steuerautomatik und UStVA

Achtung: Steuerschlüssel brauchen ein gültiges "Gültig ab"-Datum und werden andernfalls ignoriert.

Steuerschlüssel

00. Keine Steuer (0%) Automatikbuchung auf Konto:

00. Keine Steuer (0%) Automatikbuchung auf Konto:

Sonstige Einstellungen

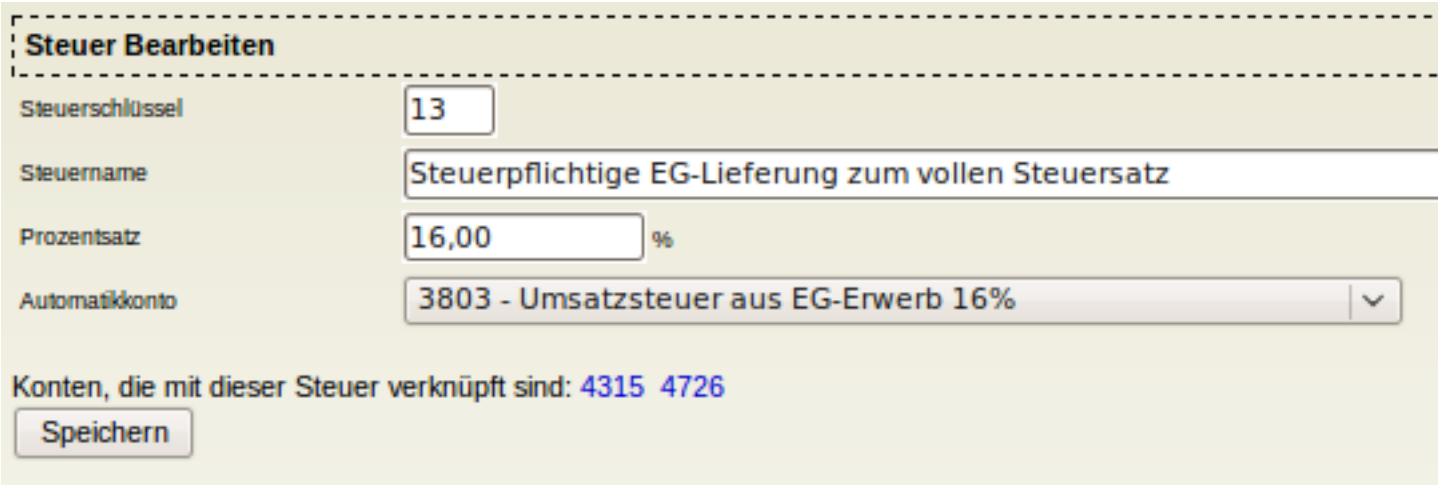
Einnahmen-/Überschussrechnung 28. Umsatzsteuerzahlungen

BWA Kein

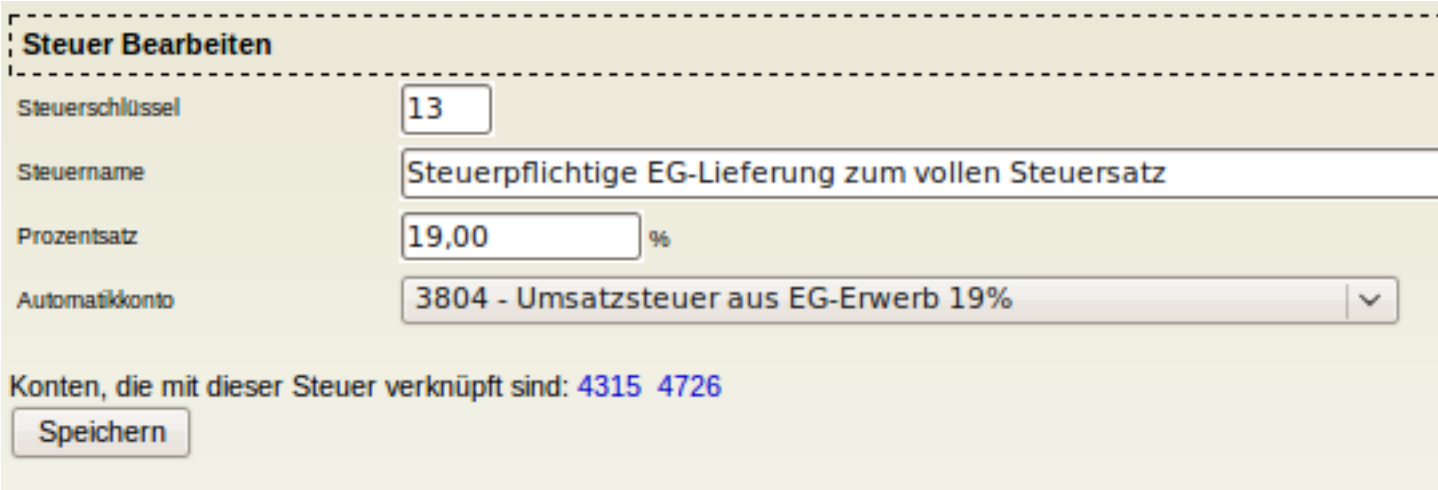
Datevexport ☐

Folgekonto Kein Gültig ab

Als Zweites muss Steuergruppe 13 für Konto 3803 angepasst werden. Dazu unter System → Steuern → Bearbeiten den Eintrag mit Steuerschlüssel 13 auswählen und ihn wie im folgenden Screenshot angezeigt anpassen.



Als Drittes wird ein neuer Eintrag mit Steuerschlüssel 13 für Konto 3804 (19%) angelegt. Dazu unter System → Steuern → Erfassen auswählen und die Werte aus dem Screenshot übernehmen.



Als Nächstes sind alle Konten anzupassen, die als Steuerautomatikkonto die 3803 haben, sodass sie ab dem 1.1.2007 auch Steuerautomatik auf 3804 bekommen. Dies betrifft in der Standardkonfiguration die Konten 4315 und 4726. Als Beispiel für 4315 müssen Sie dazu unter System → Kontenübersicht → Konten anzeigen das Konto 4315 anklicken und die Einstellungen wie im Screenshot gezeigt vornehmen.

Kontodaten bearbeiten

Grundeinstellungen

Kontonummer 4315

Beschreibung Erlös Inland stpfl.EG-Lieferung 16%/19%

Kontentyp Konto

Kontoart

Erlöskonto (I)

Buchungskonto in

☐ **Verkauf** ☐ **Einkauf** ☐ **Inventar**

In Aufklappmenü aufnehmen

Dieser Block ist nur dann gültig, wenn das Konto KEIN Buchungskonto ist, und wenn ein gültiger Steuerschlüsse

Forderungen	Verbindlichkeiten
☑ Erlöskonto	☐ Aufwand/Anlagen
☐ Zahlungseingang	☐ Zahlungsausgang
☐ Steuer	☐ Steuer

Steuerautomatik und UStVA

Achtung: Steuerschlüssel brauchen ein gültiges "Gültig ab"-Datum und werden andernfalls ignoriert.

Steuerschlüssel

13. Steuerpflichtige EG-Lieferung zum vollen Steuersatz (19%) Automatikbuchung auf Konto: 3

13. Steuerpflichtige EG-Lieferung zum vollen Steuersatz (16%) Automatikbuchung auf Konto: 3

00. Keine Steuer (0%) Automatikbuchung auf Konto:

Sonstige Einstellungen

Einnahmen-/Überschussrechnung 01. Umsatzerlöse

BWA 01. Umsatzerlöse

Datevexport ☑

Folgekonto Kein Gültig a

Als Letztes sollte die Steuerliste unter System → Steuern → Bearbeiten kontrolliert werden. Zum Vergleich der Screenshot.

Steuer

Steuerschlüssel	Steuername	Prozent	Automatikkonto	Beschreibung
0	Keine Steuer	0,00 %		
1	USt-frei	0,00 %		
2	Umsatzsteuer	7,00 %	3801	Umsatzsteuer 7%
3	Umsatzsteuer	16,00 %	3805	Umsatzsteuer 16%
3	Umsatzsteuer	19,00 %	3806	Umsatzsteuer 19 %
8	Vorsteuer	7,00 %	1401	Abziehbare Vorsteue
9	Vorsteuer	16,00 %	1405	Abziehbare Vorsteue
9	Vorsteuer	19,00 %	1406	Abziehbare Vorsteue
10	Im anderen EU-Staat steuerpflichtige Lieferung	0,00 %		
11	Steuerfreie innergem. Lieferung an Abnehmer mit Id.-Nr.	0,00 %		
12	Steuerpflichtige EG-Lieferung zum ermäßigten Steuersatz	7,00 %	3802	Umsatzsteuer aus EG
13	Steuerpflichtige EG-Lieferung zum vollen Steuersatz	16,00 %	3803	Umsatzsteuer aus EG
13	Steuerpflichtige EG-Lieferung zum vollen Steuersatz	19,00 %	3804	Umsatzsteuer aus EG
18	Steuerpflichtiger innergem. Erwerb zum ermäßigten Steuersatz	7,00 %	1402	Abziehbare Vorsteue
19	Steuerpflichtige EG-Lieferung zum vollen Steuersatz	19,00 %	1404	Abziehbare Vorsteue
19	Steuerpflichtiger innergem. Erwerb zum vollen Steuersatz	16,00 %	1403	Abziehbare Vorsteue

Erfassen

2.19. Verhalten des Bilanzberichts

Bis Version 3.0 wurde "closedto" ("Bücher schließen zum") als Grundlage für das Startdatum benutzt. Schließt man die Bücher allerdings monatsweise führt dies zu falschen Werten.

In der Mandantenkonfiguration kann man dieses Verhalten genau einstellen indem man:

- weiterhin closed_to benutzt (Default, es ändert sich nichts zu vorher)
- immer den Jahresanfang nimmt (1.1. relativ zum Stichtag)
- immer die letzte Eröffnungsbuchung als Startdatum nimmt

  - mit Jahresanfang als Alternative wenn es keine EB-Buchungen gibt

  - oder mit "alle Buchungen" als Alternative"
- mit Jahresanfang als Alternative wenn es keine EB-Buchungen gibt
- immer alle Buchungen seit Beginn der Datenbank nimmt

Folgende Hinweise zu den Optionen: Das "Bücher schließen Datum" ist sinnvoll, wenn man nur komplette Jahre schließt. Bei Wirtschaftsjahr = Kalendarjahr entspricht dies aber auch dem Jahresanfang. "Alle Buchungen" kann z.B. sinnvoll sein wenn man ohne Jahresabschluß durchbucht. Eröffnungsbuchung mit "alle Buchungen" als Fallback ist z.B. sinnvoll, wenn man am sich Anfang des zweiten Buchungsjahres befindet, und noch keinen Jahreswechsel und auch noch keine EB-Buchungen hat. Bei den Optionen mit EB-Buchungen wird vorausgesetzt, daß diese immer am 1. Tag des Wirtschaftsjahres gebucht werden. Zur Sicherheit wird das Startdatum im Bilanzbericht jetzt zusätzlich zum Stichtag mit angezeigt. Das hilft auch bei der Kontrolle für den Abgleich mit der GuV bzw. Erfolgsrechnung.

2.20. Jahresabschluss

Beim Jahresabschluss stehen 2 Methoden zur Verfügung, die in der Buchungskonfiguration der Mandantenkonfiguration eingestellt werden können.

- **Standard-Methode**.
  - Sowohl Bestandskonten wie auch Erfolgskonten werden abgeschlossen
  - Es wird ein Konto für das Verbuchen eines Jahresgewinns verwendet und ein zweites Konto für das Verbuchen eines Jahresverlusts.
  - Bei den Konten für den Jahresabschluss können bei den Standardkonten in der Mandantenkonfiguration nur Aktiva-Konten ausgewählt werden.
  - Der Jahresgewinn oder -verlust wird sowohl bei den Jahresabschlussbuchungen wie auch bei den Eröffnungsbuchungen verbucht.
- **Einfach-Methode**.
  - Nur Bestandskonten werden abgeschlossen
  - Für das Verbuchen eines Jahresgewinns oder -verlusts wird das gleiche Konto verwendet.
  - Bei den Konten für den Jahresabschluss können bei den Standardkonten in der Mandantenkonfiguration alle Konten ausgewählt werden.
  - Der Jahresgewinn oder -verlust wird nur bei den Eröffnungsbuchungen verbucht.

2.21. Erfolgsrechnung

Seit der Version 3.4.1 existiert in kivitendo der Bericht **Erfolgsrechnung**.

Die Erfolgsrechnung kann in der Mandantenkonfiguration unter Features an- oder abgeschaltet werden. Mit der Einstellung `default_manager = swiss` in der `config/kivitendo.conf` wird beim neu Erstellen einer Datenbank automatisch die Anzeige der Erfolgsrechnung im Menü Berichte ausgewählt und ersetzt dort die GUV.

Im Gegensatz zur GUV werden bei der Erfolgsrechnung sämtliche Aufwands- und Erlöskonten einzeln aufgelistet (analog zur Bilanz), sortiert nach ERTRAG und AUFWAND.

Bei den Konteneinstellungen muss bei jedem Konto, das in der Erfolgsrechnung erscheinen soll, unter `Sonstige Einstellungen/Erfolgsrechnung` entweder `01.Ertrag` oder `06.Aufwand` ausgewählt werden.

Wird bei einem Erlöskonto `06.Aufwand` ausgewählt, so wird dieses Konto als Aufwandsminderung unter AUFWAND aufgelistet.

Wird bei einem Aufwandskonto `01.Ertrag` ausgewählt, so wird dieses Konto als Ertragsminderung unter ERTRAG aufgelistet.

Soll bei einer bereits bestehenden Buchhaltung in Zukunft zusätzlich die Erfolgsrechnung als Bericht verwendet werden, so müssen die Einstellungen zu allen Erlös- und Aufwandskonten unter `Sonstige Einstellungen/Erfolgsrechnung` überprüft und allenfalls neu gesetzt werden.

2.22. Rundung in Verkaufsbelegen

In der Schweiz hat die kleinste aktuell benutzte Münze den Wert von 5 Rappen (0.05 CHF).

Auch wenn im elektronischen Zahlungsverkehr Beträge mit einer Genauigkeit von 0.01 CHF verwendet werden können, ist es trotzdem nach wie vor üblich, Rechnungen mit auf 0.05 CHF gerundeten Beträgen auszustellen.

In kivitendo kann seit der Version 3.4.1 die Einstellung für eine solche Rundung pro Mandant / Datenbank festgelegt werden.

Die Einstellung wird beim Erstellen der Datenbank bei `Genauigkeit` festgelegt. Sie kann anschliessend über das Webinterface von kivitendo nicht mehr verändert werden.

Abhängig vom Wert für `default_manager` in `config/kivitendo.conf` werden dabei folgende Werte voreingestellt:

- 0.05 (default_manager = swiss)
- 0.01 (default_manager = german)

Der Wert wird in der Datenbank in der Tabelle `defaults` in der Spalte `precision` gespeichert.

In allen Verkaufsangeboten, Verkaufsaufträgen, Verkaufsrechnungen und Verkaufsgutschriften wird der Endbetrag inkl. MWST gerundet, wenn dieser nicht der eingestellten Genauigkeit entspricht.

Beim Buchen einer Verkaufsrechnung wird der Rundungsbetrag automatisch auf die in der Mandantenkonfiguration festgelegten Standardkonten für Rundungserträge bzw. Rundungsaufwendungen gebucht.

(Die berechnete MWST wird durch den Rundungsbetrag nicht mehr verändert.)

Die in den Druckvorlagen zur Verfügung stehenden Variablen `quototal`, `ordtotal` bzw. `invtotal` enthalten den gerundeten Betrag.

Achtung: Werden Verkaufsbelege in anderen Währungen als der Standardwährung erstellt, so muss in kivitendo ab Version 3.4.1 die Genauigkeit 0.01 verwendet werden.

Das heisst, Firmen in der Schweiz, die teilweise Verkaufsrechnungen in Euro oder anderen Währungen erstellen wollen, müssen beim Erstellen der Datenbank als Genauigkeit 0.01 wählen und können zur Zeit die 5er Rundung noch nicht nutzen.

2.23. Einstellungen pro Mandant

Einige Einstellungen können von einem Benutzer mit dem Recht "Administration (Für die Verwaltung der aktuellen Instanz aus einem Userlogin heraus)" gemacht werden. Diese Einstellungen sind dann für die aktuellen Mandanten-Datenbank gültig. Die Einstellungen sind unter System → Mandantenkonfiguration erreichbar.

Bitte beachten Sie die Hinweise zu den einzelnen Einstellungen. Einige Einstellungen sollten nicht ohne Weiteres im laufenden Betrieb geändert werden (siehe auch Bemerkungen zu Bestandsmethode).

Die Einstellungen `show_bestbefore` und `payments_changeable` aus dem Abschnitt `features` und die Einstellungen im Abschnitt `datev_check` (sofern schon vorhanden) der kivitendo-Konfigurationsdatei werden bei einem Datenbankupdate einer älteren Version automatisch übernommen. Diese Einträge können danach aus der Konfigurationsdatei entfernt werden.

2.24. kivitendo ERP verwenden

Nach erfolgreicher Installation ist der Loginbildschirm unter folgender URL erreichbar:

http://localhost/kivitendo-erp/login.pl

Die Administrationsseite erreichen Sie unter:

http://localhost/kivitendo-erp/controller.pl?action=Admin/login

3
Features und Funktionen

3.1. Wiederkehrende Rechnungen

3.1.1. Einführung

Wiederkehrende Rechnungen werden als normale Aufträge definiert und konfiguriert, mit allen dazugehörigen Kunden- und Artikelangaben. Die konfigurierten Aufträge werden später automatisch in Rechnungen umgewandelt, so als ob man den Workflow benutzen würde, und auch die Auftragsnummer wird übernommen, sodass alle wiederkehrenden Rechnungen, die aus einem Auftrag erstellt wurden, später leicht wiederzufinden sind.

3.1.2. Konfiguration

Um einen Auftrag für wiederkehrende Rechnung zu konfigurieren, findet sich beim Bearbeiten des Auftrags ein neuer Knopf "Konfigurieren", der ein neues Fenster öffnet, in dem man die nötigen Parameter einstellen kann. Hinter dem Knopf wird außerdem noch angezeigt, ob der Auftrag als wiederkehrende Rechnung konfiguriert ist oder nicht.

Folgende Parameter kann man konfigurieren:

Status
: Bei aktiven Rechnungen wird automatisch eine Rechnung erstellt, wenn die Periodizität erreicht ist (z.B. am Anfang eines neuen Monats).

  Ist ein Auftrag nicht aktiv, so werden für ihn auch keine wiederkehrenden Rechnungen erzeugt. Stellt man nach längerer nicht-aktiver Zeit einen Auftrag wieder auf aktiv, wird beim nächsten Periodenwechsel für alle Perioden, seit der letzten aktiven Periode, jeweils eine Rechnung erstellt. Möchte man dies verhindern, muss man vorher das Startdatum neu setzen.

  Für gekündigte Aufträge werden nie mehr Rechnungen erstellt. Man kann sich diese Aufträge aber gesondert in den Berichten anzeigen lassen.

Periodizität
: Ob monatlich, quartalsweise oder jährlich auf neue Rechnungen überprüft werden soll. Für jede Periode seit dem Startdatum wird überprüft, ob für die Periode (beginnend immer mit dem ersten Tag der Periode) schon eine Rechnung erstellt wurde. Unter Umständen können bei einem Startdatum in der Vergangenheit gleich mehrere Rechnungen erstellt werden.

Buchen auf
: Das Forderungskonto, in der Regel "Forderungen aus Lieferungen und Leistungen". Das Gegenkonto ergibt sich aus den Buchungsgruppen der betreffenden Waren.

Startdatum
: ab welchem Datum auf Rechnungserstellung geprüft werden soll

Enddatum
: ab wann keine Rechnungen mehr erstellt werden sollen

Automatische Verlängerung um x Monate
: Sollen die wiederkehrenden Rechnungen bei Erreichen des eingetragenen Enddatums weiterhin erstellt werden, so kann man hier die Anzahl der Monate eingeben, um die das Enddatum automatisch nach hinten geschoben wird.

Drucken
: Sind Drucker konfiguriert, so kann man sich die erstellten Rechnungen auch gleich ausdrucken lassen.

Nach Erstellung der Rechnungen kann eine E-Mail mit Informationen zu den erstellten Rechnungen verschickt werden. Konfiguriert wird dies in der Konfigurationsdatei `config/kivitendo.conf` im Abschnitt `[periodic_invoices]`.

3.1.3. Spezielle Variablen

Um die erzeugten Rechnungen individualisieren zu können, werden beim Umwandeln des Auftrags in eine Rechnung einige speziell formatierte Variablen durch für die jeweils aktuelle Abrechnungsperiode gültigen Werte ersetzt. Damit ist es möglich, z.B. den Abrechnungszeitraum explizit auszuweisen. Eine Variable hat dabei die Syntax `<%variablenname%>`.

Sofern es sich um eine Datumsvariable handelt, kann das Ausgabeformat weiter bestimmt werden, indem an den Variablennamen Formatoptionen angehängt werden. Die Syntax sieht dabei wie folgt aus: `<%variablenname FORMAT=Formatinformation%>`. Die zur verfügung stehenden Formatinformationen werden unten genauer beschrieben.

Diese Variablen können auch beim automatischen Versand der erzeugten Rechnungen per E-Mail genutzt werden, indem sie in den Feldern für den Betreff oder die Nachricht verwendet werden.

Diese Variablen werden in den folgenden Elementen des Auftrags ersetzt:

- Bemerkungen
- Interne Bemerkungen
- Vorgangsbezeichnung
- In den Beschreibungs- und Langtextfeldern aller Positionen

Die zur Verfügung stehenden Variablen sind die Folgenden:

`<%current_quarter%>`, `<%previous_quarter%>`, `<%next_quarter%>`
: Aktuelles, vorheriges und nächstes Quartal als Zahl zwischen `1` und `4`.

`<%current_month%>`, `<%previous_month%>`, `<%next_month%>`
: Aktueller, vorheriger und nächster Monat als Zahl zwischen `1` und `12`.

`<%current_month_long%>`, `<%previous_month_long%>`, `<%next_month_long%>`
: Aktueller, vorheriger und nächster Monat als Name (`Januar`, `Februar` etc.).

`<%current_year%>`, `<%previous_year%>`, `<%next_year%>`
: Aktuelles, vorheriges und nächstes Jahr als vierstellige Jahreszahl (`2013` etc.).

`<%period_start_date%>`, `<%period_end_date%>`
: Formatiertes Datum des ersten und letzten Tages im Abrechnungszeitraum (z.B. bei quartalsweiser Abrechnung und im ersten Quartal von 2013 wären dies der `01.01.2013` und `31.03.2013`).

Die invidiuellen Formatinformationen bestehen aus Paaren von Prozentzeichen und einem Buchstaben, welche beide zusammen durch den dazugehörigen Wert ersetzt werden. So wird z.B. `%Y` durch das viertstellige Jahr ersetzt. Alle möglichen Platzhalter sind:

`%a`
: Der abgekürzte Wochentagsname.

`%A`
: Der ausgeschriebene Wochentagsname.

`%b`
: Der abgekürzte Monatsname.

`%B`
: Der ausgeschriebene Monatsname.

`%C`
Das Jahrhundert (Jahr/100) als eine zweistellige Zahl.

`%d`
Der Monatstag als Zahl zwischen 01 und 31.

`%D`
Entspricht %m/%d/%y (amerikanisches Datumsformat).

`%e`
Wie %d (Monatstag als Zahl zwischen 1 und 31), allerdings werden führende Nullen durch Leerzeichen ersetzt.

`%F`
Entspricht %Y-%m-%d (das ISO-8601-Datumsformat).

`%j`
Der Tag im Jahr als Zahl zwischen 001 und 366 inklusive.

`%m`
Der Monat als Zahl zwischen 01 und 12 inklusive.

`%u`
Der Wochentag als Zahl zwischen 1 und 7 inklusive, wobei die 1 dem Montag entspricht.

`%U`
Die Wochennummer als Zahl zwischen 00 und 53 inklusive, wobei der erste Sonntag im Jahr das Startdatum von Woche 01 ist.

`%V`
Die ISO-8601:1988-Wochennummer als Zahl zwischen 01 und 53 inklusive, wobei Woche 01 die erste Woche, von der mindestens vier Tage im Jahr liegen; Montag ist erster Tag der Woche.

`%w`
Der Wochentag als Zahl zwischen 0 und 6 inklusive, wobei die 0 dem Sonntag entspricht.

`%W`
Die Wochennummer als Zahl zwischen 00 und 53 inklusive, wobei der erste Montag im Jahr das Startdatum von Woche 01 ist.

`%y`
Das Jahr als zweistellige Zahl zwischen 00 und 99 inklusive.

`%Y`
Das Jahr als vierstellige Zahl.

`%%`
Das Prozentzeichen selber.

Anwendungsbeispiel für die Ausgabe, von welchem Monat und Jahr bis zu welchem Monat und Jahr die aktuelle Abrechnungsperiode dauert: `Abrechnungszeitrum: <%period_start_date FORMAT=%m/%Y%> bis <%period_end_date FORMAT=%m/%Y%>`

Beim automatischen Versand der Rechnugen via E-Mail können neben diesen speziellen Variablen auch einige Eigenschaften der Rechnung selber als Variablen im Betreff & dem Text der E-Mails genutzt werden. Beispiele sind `<%invnumber%>` für die Rechnungsnummber oder `<transaction_description%>` für die Vorgangsbezeichnung. Diese Variablen stehen beim Erzeugen der Rechnung logischerweise noch nicht zur Verfügung.

3.1.4. Auflisten

Unter Verkauf->Berichte->Aufträge finden sich zwei neue Checkboxen, "Wiederkehrende Rechnungen aktiv" und "Wiederkehrende Rechnungen inaktiv", mit denen man sich einen Überglick über die wiederkehrenden Rechnungen verschaffen kann.

3.1.5. Erzeugung der eigentlichen Rechnungen

Die zeitliche und periodische Überprüfung, ob eine wiederkehrende Rechnung automatisch erstellt werden soll, geschieht durch den Task-Server, einen externen Dienst, der automatisch beim Start des Servers gestartet werden sollte.

3.1.6. Erste Rechnung für aktuellen Monat erstellen

Will man im laufenden Monat eine monatlich wiederkehrende Rechnung inkl. des laufenden Monats starten, stellt man das Startdatum auf den Monatsanfang und wartet ein paar Minuten, bis der Task-Server den neu konfigurierten Auftrag erkennt und daraus eine Rechnung generiert hat. Alternativ setzt man das Startdatum auf den Monatsersten des Folgemonats und erstellt die erste Rechnung direkt manuell über den Workflow.

3.2. Zahlungsverkehr über Bankkonten

kivitendo kann Banküberweisungen in den Formaten MT940, camt.053 und CSV einlesen, um automatisch Vorschläge zum Verbuchen von Bankbewegungen zu offene Verkaufs- wie Einkaufsrechungen zu erzeugen. Diese Vorschläge werden nach manueller Kontrolle verbucht. Ferner kann kivitendo Anweisungen an die Bank für SEPA-Überweisungen und SEPA-Bankeinzüge erzeugen, welche in Form von SEPA XML Dateien im Onlinebanking oder Bankingprogramm eingespielt werden können.

Eine Nähere Beschreibung findet sich im Abschnitt Zahlungsverkehr unter:

https://howto.kivitendo.de/course/view.php?id=35 [1]

3.2.1. SEPA-Überweisungen

Sowohl einzelne Überweisungen pro Beleg als auch Sammelüberweisungen gruppiert nach Lieferant werden unterstützt. Beim Rückimport von der Bank über das camt.053, MT940 oder CSV Verfahren werden diese Überweisung direkt beim Import automatisch verbucht, falls die SEPA EndToEndID und die Überweisungssumme identisch zu einer per SEPA-Export erzeugen Überweisung sind. Damit entfällt in diesem Fall die manuelle Kontrolle beim Verbuchen der Bankbewegung.

Lieferantenrechnungen können mit Gutschriften verrechnet werden. Dabei ist mit Gutschrift eine Kreditorenbuchung mit negativem Betrag gemeint. Damit Gutschriften in der Oberfläche unter *Überweisung via SEPA XML vorbereiten* angezeigt werden, sind folgende Einstelloptionen in der Mandantenkonfiguration zu setzen:

- Gutschriften beim SEPA-Lauf verrechnen: Ja (Tab SEPA/Bank)
- Veränderbarkeit von Zahlungen: niemals (Tab Buchungskonfiguration)
- Standardkonto für durchlaufenden Posten: z.B. Konto 1370 im SKR03 (Tab Standardkonten)

Eine explizite Freigabe von SEPA-Überweisungen für alle Einkaufsbelege lässt sich in der Mandantenkonfiguration aktivieren.

3.2.2. Kontoauszug Verbuchen mit Buchungsvorlagen und direktem Verbuchen

Beim Verbuchen von Kontoauszügen werden zu jeder Banktransaktion drei Optionen angeboten:

- »Rechnung zuweisen«
- »Buchung erstellen«: Auswahl und Laden einer Buchungsvorlage
- »NI optimierte Buchung«:[2] Die passende Dialogbuchungsvorlage wird direkt aufgerufen.

[1] https://howto.kivitendo.de/course/view.php?id=35
[2] NI steht für natürliche Intelligenz

Existiert beim Kontoauszugsimport eine Dialogbuchungsvorlage mit Beschreibungstext (inkl. Variablenersetzung), welcher zu dem Verwendungszweck einer Banktransaktionen passt, dann wird die Banktransaktion direkt beim Importieren des Kontoauszuges automatisch verbucht. Zur Kontrolle lässt sich das automatische Verbuchen beim Kontoauszugsimport deaktivieren.

3.3. Dokumentenvorlagen und verfügbare Variablen

3.3.1. Einführung

Dies ist eine Auflistung der Standard-Dokumentenvorlagen und aller zur Bearbeitung verfügbaren Variablen. Eine Variable wird in einer Vorlage durch ihren Inhalt ersetzt, wenn sie in der Form `<%variablenname%>` verwendet wird. Für LaTeX- und HTML-Vorlagen kann man die Form dieser Tags auch verändern (siehe Anfang und Ende der Tags verändern [58]).

kivitendo unterstützt LaTeX-, HTML- und OpenDocument-Vorlagen. Sofern es nicht ausdrücklich eingeschränkt wird, gilt das im Folgenden gesagte für alle Vorlagenarten.

Insgesamt sind technisch gesehen eine ganze Menge mehr Variablen verfügbar als hier aufgelistet werden. Die meisten davon können allerdings innerhalb einer solchen Vorlage nicht sinnvoll verwendet werden. Wenn eine Auflistung dieser Variablen gewollt ist, so kann diese wie folgt erhalten werden:

- `SL/Form.pm` öffnen und am Anfang die Zeile "**use Data::Dumper;**" einfügen.
- In `Form.pm` die Funktion `parse_template` suchen und hier die Zeile **print(STDERR Dumper($self));** einfügen.
- Einmal per Browser die gewünschte Vorlage "benutzen", z.B. ein PDF für eine Rechnung erzeugen.
- Im `error.log` Apache steht die Ausgabe der Variablen `$self` in der Form `'key' => 'value',`. Alle `key`s sind verfügbar.

3.3.2. Variablen ausgeben

Um eine Variable auszugeben, müssen sie einfach nur zwischen die Tags geschrieben werden, also z.B. `<%variablenname%>`.

Optional kann man auch mit Leerzeichen getrennte Flags angeben, die man aber nur selten brauchen wird. Die Syntax sieht also so aus: `<%variablenname FLAG1 FLAG2%>`. Momentan werden die folgenden Flags unterstützt:

- `NOFORMAT` gilt nur für Zahlenwerte und gibt den Wert ohne Formatierung, also ohne Tausendertrennzeichen mit mit einem Punkt als Dezimaltrennzeichen aus. Nützlich z.B., wenn damit in der Vorlage z.B. von LaTeX gerechnet werden soll.
- `NOESCAPE` unterdrückt das Escapen von Sonderzeichen für die Vorlagensprache. Wenn also in einer Variablen bereits gültiger LaTeX-Code steht und dieser von LaTeX auch ausgewertet und nicht wortwörtlich angezeigt werden soll, so ist dieses Flag sinnvoll.

Beispiel:

```
<%quototal NOFORMAT%>
```

3.3.3. Verwendung in Druckbefehlen

In der Admininstration können Drucker definiert werden. Auch im dort eingebbaren Druckbefehl können die hier aufgelisteten Variablen und Kontrollstrukturen verwendet werden. Ihr Inhalt wird dabei nach den Regeln der gängigen Shells formatiert, sodass Sonderzeichen wie `` `...` `` nicht zu unerwünschtem Verhalten führen.

Dies erlaubt z.B. die Definition eines Faxes als Druckerbefehl, für das die Telefonnummer eines Ansprechpartners als Teil der Kommandozeile verwendet wird. Für ein fiktives Kommando könnte das z.B. wie folgt aussehen:

```
send_fax --number <%if cp_phone2%><%cp_phone2%><%else%><%cp_phone1%><%end%>
```

3.3.4. Anfang und Ende der Tags verändern

Der Standardstil für Tags sieht vor, dass ein Tag mit dem Kleinerzeichen und einem Prozentzeichen beginnt und mit dem Prozentzeichen und dem Größerzeichen endet, beispielsweise `<%customer%>`. Da diese Form aber z.B. in LaTeX zu Problemen führen kann, weil das Prozentzeichen dort Kommentare einleitet, kann pro HTML- oder LaTeX-Dokumentenvorlage der Stil umgestellt werden.

Dazu werden in die Datei Zeilen geschrieben, die mit dem für das Format gültigen Kommentarzeichen anfangen, dann `config:` enthalten, die entsprechende Option setzen und bei HTML-Dokumentenvorlagen mit dem Kommentarendzeichen enden. Beispiel für LaTeX:

```
% config: tag-style=($ $)
```

Dies würde kivitendo dazu veranlassen, Variablen zu ersetzen, wenn sie wie folgt aussehen: `($customer$)`. Das äquivalente Beispiel für HTML-Dokumentenvorlagen sieht so aus:

```
<!-- config: tag-style=($ $) -->
```

3.3.5. Zuordnung von den Dateinamen zu den Funktionen

Diese folgende kurze Auflistung zeigt, welche Vorlage bei welcher Funktion ausgelesen wird. Dabei ist die Dateiendung `".ext"` geeignet zu ersetzen: `".tex"` für LaTeX-Vorlagen und `".odt"` für OpenDocument-Vorlagen.

`bin_list.ext`
: Lagerliste

`check.ext`
: ?

`invoice.ext`
: Rechnung

`packing_list.ext`
: Packliste

`pick_list.ext`
: Sammelliste

`purchase_delivery_order.ext`
: Lieferschein (Einkauf)

`purcharse_order.ext`
: Bestellung an Lieferanten

`request_quotation.ext`
: Anfrage an Lieferanten

`sales_delivery_order.ext`
: Lieferschein (Verkauf)

`sales_order.ext`
: Bestellung

`sales_quotation.ext`
: Angebot an Kunden

`zahlungserinnerung.ext`
: Mahnung (Dateiname im Programm konfigurierbar)

`zahlungserinnerung_invoice.ext`
: Rechnung über Mahngebühren (Dateiname im Programm konfigurierbar)

3.3.6. Sprache, Drucker und E-Mail

Angeforderte Sprache und Druckerkürzel in den Dateinamen mit eingearbeitet. So wird aus der Vorlage `sales_order.ext` bei Sprache `de` und Druckerkürzel `lpr2` der Vorlagenname `sales_order_de_lpr2.ext`. Zusätzlich können für E-Mails andere Vorlagen erstellt werden, diese bekommen dann noch das Kürzel `_email`, der vollständige Vorlagenname wäre dann `sales_order_email_de_lpr2.ext`. In allen Fällen kann eine Standarddatei `default.ext` hinterlegt werden. Diese wird verwendet, wenn keine der anderen Varianten gefunden wird.

Die vollständige Suchreihenfolge für einen Verkaufsauftrag mit der Sprache "de" und dem Drucker "lpr2", der per E-Mail im Format PDF verschickt wird, ist:

1. `sales_order_email_de_lpr2.tex`
2. `sales_order_de_lpr2.tex`
3. `sales_order.tex`
4. `default.tex`

Die kurzen Varianten dieser Vorlagentitel müssen dann entweder Standardwerte anzeigen, oder die angeforderten Werte selbst auswerten, siehe dazu Metainformationen zur angeforderten Vorlage [59].

Ländernamen für Berichte und Belegdruck lassen sich einstellen im Menü System → Sprachen und Übersetzungen → Ländernamen.

3.3.7. Allgemeine Variablen, die in allen Vorlagen vorhanden sind

3.3.7.1. Metainformationen zur angeforderten Vorlage

Diese Variablen liefern Informationen darüber welche Variante einer Vorlage der Benutzer angefragt hat. Sie sind nützlich für Vorlagenautoren, die aus einer zentralen Layoutvorlage die einzelnen Formulare einbinden möchten.

`template_meta.formname`
: Basisname der Vorlage. Identisch mit der Zurordnung zu den Dateinamen ohne die Erweiterung. Ein Verkaufsauftrag enthält hier `sales_order`.

`template_meta.language.description`
: Beschreibung der verwendeten Sprache

`template_meta.language.template_code`
: Vorlagenkürzel der verwendeten Sprache, identisch mit dem Kürzel das im Dateinamen verwendetet wird.

`template_meta.language.output_numberformat`
: Zahlenformat der verwendeten Sprache in der Form "`1.000,00`". Experimentell! Nur interessant für Vorlagen die mit unformatierten Werten arbeiten.

`template_meta.language.output_dateformat`
: Datumsformat der verwendeten Sprache in der Form "`dd.mm.yyyy`". Experimentell! Nur interessant für Vorlagen die mit unformatierten Werten arbeiten.

`template_meta.format`
: Das angeforderte Format. Kann im Moment die Werte `pdf`, `postscript`, `html`, `opendocument`, `opendocument_pdf` und `excel` enthalten.

`template_meta.extension`
: Dateierweiterung, wie im Dateinamen. Wird aus `format` entschieden.

`template_meta.media`
: Ausgabemedium. Kann zur Zeit die Werte `screen` für Bildschirm, `email` für E-Mail (triggert das `_email` Kürzel im Dateinamen), `printer` für Drucker, und `queue` für Warteschlange enthalten.

`template_meta.printer.description`
: Beschreibung des ausgewählten Druckers

`template_meta.printer.template_code`
: Vorlagenürzel des ausgewählten Druckers, identisch mit dem Kürzel das im Dateinamen verwendetet wird.

`template_meta.tmpfile`
: Datei-Prefix für temporäre Dateien.

3.3.7.2. Stammdaten von Kunden und Lieferanten

`account_number`
: Kontonummer

`bank`
: Name der Bank

`bank_code`
: Bankleitzahl

`bic`
: Bank-Identifikations-Code (Bank Identifier Code, BIC)

`business`
: Kunden-/Lieferantentyp

`city`
: Stadt

`contact`
: Kontakt

`country`
: Land

`c_vendor_id`
: Lieferantennummer beim Kunden (nur Kunden)

`v_customer_id`
: Kundennummer beim Lieferanten (nur Lieferanten)

`cp_email`
: Email des Ansprechpartners

`cp_givenname`
: Vorname des Ansprechpartners

`cp_greeting`
: Anrede des Ansprechpartners

`cp_name`
: Name des Ansprechpartners

`cp_phone1`
: Telefonnummer 1 des Ansprechpartners

`cp_phone2`
: Telefonnummer 2 des Ansprechpartners

`cp_title`
: Titel des Ansprechpartners

`creditlimit`
: Kreditlimit

`customeremail`
: Email des Kunden; nur für Kunden

`customerfax`
: Faxnummer des Kunden; nur für Kunden

`customernotes`
: Bemerkungen beim Kunden; nur für Kunden

`customernumber`
: Kundennummer; nur für Kunden

`customerphone`
: Telefonnummer des Kunden; nur für Kunden

`discount`
: Rabatt

`email`
: Emailadresse

`fax`
: Faxnummer

`gln`
: GLN (Globale Lokationsnummer)

`greeting`
: Anrede

`homepage`
: Homepage

`iban`
: Internationale Kontonummer (International Bank Account Number, IBAN)

`language`
: Sprache

`name`
: Firmenname

`natural_person`
: Flag "natürliche Person"; Siehe auch Hinweise zur Anrede [75]

`payment_description`
: Name der Zahlart

`payment_terms`
: Zahlungskonditionen

`phone`
Telefonnummer

`shiptocity`
Stadt (Lieferadresse) *

`shiptocontact`
Kontakt (Lieferadresse) *

`shiptocountry`
Land (Lieferadresse) *

`shiptodepartment_1`
Abteilung 1 (Lieferadresse) *

`shiptodepartment_2`
Abteilung 2 (Lieferadresse) *

`shiptoemail`
Email (Lieferadresse) *

`shiptofax`
Fax (Lieferadresse) *

`shiptogln`
GLN (Globale Lokationsnummer) (Lieferadresse) *

`shiptoname`
Firmenname (Lieferadresse) *

`shiptophone`
Telefonnummer (Lieferadresse) *

`shiptostreet`
Straße und Hausnummer (Lieferadresse) *

`shiptozipcode`
Postleitzahl (Lieferadresse) *

`street`
Straße und Hausnummer

`taxnumber`
Steuernummer

`ustid`
Umsatzsteuer-Identifikationsnummer

`vendoremail`
Email des Lieferanten; nur für Lieferanten

`vendorfax`
Faxnummer des Lieferanten; nur für Lieferanten

`vendornotes`
Bemerkungen beim Lieferanten; nur für Lieferanten

`vendornumber`
Lieferantennummer; nur für Lieferanten

`vendorphone`
Telefonnummer des Lieferanten; nur für Lieferanten

`zipcode`
: Postleitzahl

Anmerkung

Anmerkung: Sind die `shipto*`-Felder in den Stammdaten nicht eingetragen, so haben die Variablen `shipto*` den gleichen Wert wie die die entsprechenden Variablen der Lieferdaten. Das bedeutet, dass sich einige `shipto*`-Variablen so nicht in den Stammdaten wiederfinden sondern schlicht Kopien der Lieferdatenvariablen sind (z.B. `shiptocontact`).

3.3.7.3. Informationen über den Bearbeiter

`employee_address`
: Adressfeld

`employee_businessnumber`
: Firmennummer

`employee_company`
: Firmenname

`employee_co_ustid`
: Usatzsteuer-Identifikationsnummer

`employee_duns`
: DUNS-Nummer

`employee_email`
: Email

`employee_fax`
: Fax

`employee_name`
: voller Name

`employee_signature`
: Signatur

`employee_taxnumber`
: Steuernummer

`employee_tel`
: Telefonnummer

3.3.7.4. Informationen über den Verkäufer

`salesman_address`
: Adressfeld

`salesman_businessnumber`
: Firmennummer

`salesman_company`
: Firmenname

`salesman_co_ustid`
: Usatzsteuer-Identifikationsnummer

`salesman_duns`
: DUNS-Nummer

`salesman_email`
: Email

`salesman_fax`
: Fax

`salesman_name`
: voller Name

`salesman_signature`
: Signatur

`salesman_taxnumber`
: Steuernummer

`salesman_tel`
: Telefonnummer

3.3.7.5. Variablen für die einzelnen Steuern

`tax`
: Steuer

`taxbase`
: zu versteuernder Betrag

`taxdescription`
: Name der Steuer

`taxrate`
: Steuersatz

3.3.7.6. Variablen für Lieferbedingungen

`delivery_term`
: Datenbank-Objekt der Lieferbedingung

`delivery_term.description`
: Beschreibung der Lieferbedingung

`delivery_term.description_long`
: Langtext bzw. übersetzter Langtext der Lieferbedingung

3.3.7.7. Informationen über abweichende Rechnungsadressen (nur Verkaufsbelege)

Abweichende Rechnungsadressen gibt es nur in Verkaufsbelegen. Die entsprechenden Variablen sind nur dann mit Inhalt gefüllt, wenn im Beleg eine abweichende Rechnungsadresse ausgewählt wurde. Ob eine Adresse überhaupt ausgewählt wurde, kann über die Variable `billing_address_id` getestet werden, die die Datenbank-ID der abweichenden Rechnungsadresse enthält, wenn eine ausgewählt ist.

Die Variablennamen starten alle mit dem Präfix `billing_address_` und heißen anschließend so, wie ihre Pendants aus der Standard-Rechnungsadresse des Kunden. Beispiel: die Postleitzahl, die in der normalen Rechnungsadresse in `zipcode` steht, steht für die abweichende Rechnungsadresse in `billing_address_zipcode`.

Die folgenden Variablen stehen so zur Verfügung: `billing_address_name`, `billing_address_department_1`, `billing_address_department_2`, `billing_address_contact`, `billing_address_street`, `billing_address_zipcode`, `billing_address_city`, `billing_address_country`, `billing_address_gln`, `billing_address_ustid`, `billing_address_taxnumber`, `billing_address_email`, `billing_address_phone` und `billing_address_fax`.

3.3.8. Variablen in Rechnungen

3.3.8.1. Allgemeine Variablen

`creditremaining`
: Verbleibender Kredit

`currency`
: Währung

`cusordnumber`
: Bestellnummer beim Kunden

`deliverydate`
: Lieferdatum

`duedate`
: Fälligkeitsdatum

`globalprojectnumber`
: Projektnummer des ganzen Beleges

`globalprojectdescription`
: Projekbeschreibung des ganzen Beleges

`intnotes`
: Interne Bemerkungen

`invdate`
: Rechnungsdatum

`invnumber`
: Rechnungsnummer

`invtotal`
: gesamter Rechnungsbetrag

`notes`
: Bemerkungen der Rechnung

`orddate`
: Auftragsdatum

`ordnumber`
: Auftragsnummer, wenn die Rechnung aus einem Auftrag erstellt wurde

`payment_description`
: Name der Zahlart

`payment_terms`
: Zahlungskonditionen

`quodate`
: Angebotsdatum

`quonumber`
: Angebotsnummer

`rounding`
: Betrag, um den `invtotal` gerundet wurde (kann positiv oder negativ sein)

`shippingpoint`
: Versandort

`shipvia`
: Transportmittel

`subtotal`
: Zwischensumme aller Posten ohne Steuern

`total`
: Restsumme der Rechnung (Summe abzüglich bereits bezahlter Posten)

`transaction_description`
: Vorgangsbezeichnung

`transdate`
: Auftragsdatum wenn die Rechnung aus einem Auftrag erstellt wurde

3.3.8.2. Variablen für die schweizer QR-Rechnung

Diese Variablen können mit dem LaTeX Modul qrbill verwendet werden: https://ctan.org/pkg/qrbill?lang=de

Für die Erstellung von QR-Rechnungen mit OpenDocument Vorlagen siehe: Abschnitt 2.15, „OpenDocument-Vorlagen“ [41]

qrbill_iban
: IBAN/QR-IBAN des Rechnungsstellers, aus *System → Bankkonten*

qrbill_biller_countrycode
: Länderkürzel des Rechnungsstellers gem. ISO 3166, aus *Mandantenkonfiguration → Firmenname und -adresse*

qrbill_customer_countrycode
: Länderkürzel des Rechnungsempfängers gem. ISO 3166, aus der jeweiligen Rechnung

qrbill_amount
: Betrag für die QR-Rechnung (Zahl ohne Tausendertrennzeichen mit zwei Nachkommastellen), entsprechend total

qr_reference
: QR-Referenz der jeweiligen Rechnung, sofern in der *Mandantenkonfiguration → Features → Variante QR-IBAN mit QR-Referenz erzeugen* aktiviert ist

3.3.8.3. Variablen für jeden Posten auf der Rechnung

`bin`
: Stellage

`description`
: Artikelbeschreibung

`cusordnumber_oe`
: Bestellnummer des Kunden aus dem Auftrag, aus dem der Posten ursprünglich stammt (nur Verkauf)

`discount`
: Rabatt als Betrag

`discount_sub`
: Zwischensumme mit Rabatt

`donumber_do`
: Lieferscheinnummer des Lieferscheins, aus dem die Position ursprünglich stammt, wenn die Rechnung im Rahmen des Workflows aus einem Lieferschein erstellt wurde.

`drawing`
: Zeichnung

`ean`
: EAN-Code

`image`
: Grafik

`linetotal`
: Zeilensumme (Anzahl * Einzelpreis)

`longdescription`
: Langtext, vorbelegt mit dem Feld Bemerkungen der entsprechenden Ware

`microfiche`
: Mikrofilm

`netprice`
: Alternative zu `sellprice`, aber `netprice` entspricht dem effektiven Einzelpreis und beinhaltet Zeilenrabatt und Preisfaktor. `netprice` wird rückgerechnet aus Zeilensumme / Menge. Diese Variable ist nützlich, wenn man den gewährten Rabatt in der Druckvorlage nicht anzeigen möchte, aber Menge * Einzelpreis trotzdem die angezeigte Zeilensumme ergeben soll. `netprice` hat nichts mit Netto/Brutto im Sinne von Steuern zu tun.

`nodiscount_linetotal`
: Zeilensumme ohne Rabatt

`nodiscount_sub`
: Zwischensumme ohne Rabatt

`number`
: Artikelnummer

`ordnumber_oe`
: Auftragsnummer des Originalauftrags, aus dem der Posten ursprünglich stammt. Nützlich, wenn die Rechnung aus mehreren Lieferscheinen zusammengefasst wurde, oder wenn zwischendurch eine Sammelauftrag aus mehreren Aufträgen erstellt wurde. In letzterem Fall wird die unsprüngliche Auftragsnummer angezeigt.

`p_discount`
: Rabatt in Prozent

`partnotes`
: Die beim Artikel gespeicherten Bemerkungen

`partsgroup`
: Warengruppe

`price_factor`
: Der Preisfaktor als Zahl, sofern einer eingestellt ist

`price_factor_name`
: Der Name des Preisfaktors, sofern einer eingestellt ist

`projectnumber`
: Projektnummer

`projectdescription`
: Projektbeschreibung

`qty`
: Anzahl

`reqdate`
: Lieferdatum

`runningnumber`
: Position auf der Rechnung (1, 2, 3...)

`sellprice`
: Verkaufspreis

`serialnumber`
: Seriennummer

`tax_rate`
: Steuersatz

`transdate_do`
: Datum des Lieferscheins, wenn die Rechnung im Rahmen des Workflows aus einem Lieferschein stammte.

`transdate_oe`
: Datum des Auftrags, wenn die Rechnung im Rahmen des Workflows aus einem Auftrag erstellt wurde. Wenn es Sammelaufträge gab wird das Datum des ursprünglichen Auftrags genommen.

`transdate_quo`
: Datum des Angebots, wenn die Position im Rahmen des Workflows aus einem Angebot stammte.

`unit`
: Einheit

`weight`
: Gewicht

Für jeden Posten gibt es ein Unterarray mit den Informationen über Lieferanten und Lieferantenartikelnummer, Kunde und Kundenartikelnummer und Kunden- bzw. Lieferantentyp und zugehöriger Artikelnummer mit Beschreibung und Langtext. Diese müssen jeweils mit einer `foreach`-Schleife ausgegeben werden, da für jeden Artikel mehrere Lieferanten- und Kundeninformationen bzw. kunden- bzw. lieferantenspezifische Informationen hinterlegt sein können. Die Variablen dafür lauten:

`make`
: Lieferant

`model`
: Lieferantenartikelnummer

`mm_part_description`
: Lieferantenartikelbeschreibung

`mm_part_longdescription`
: Lieferantenartikelbeschreibung (Langtext)

`customer_make`
: Kunde

`customer_model`
: Kundenartikelnummer

`cm_part_description`
: Kundenartikelbeschreibung

`cm_part_longdescription`
: Kundenartikelbeschreibung (Langtext)

`business_make`
: Kunden-/Lieferantentyp

`business_model`
: Kunden-/Lieferantentyp-spezifische Artikelnummer

`bm_part_description`
: Kunden-/Lieferantentyp-spezifische Artikelbeschreibung

`bm_part_longdescription`
: Kunden-/Lieferantentyp-spezifische Artikelbeschreibung (Langtext)

3.3.8.4. Benutzerdefinierte Variablen für jeden Posten auf der Rechnung

Für jeden Posten stehen auch die benutzerdefinierten Variablen zum Artikel zur Verfügung. Ihre Namen bestehen aus dem Präfix `ic_cvar_` und dem vom Benutzer festgelegten Variablennamen.

Ebenso stehen die benutzerdefinierten Variablen zum positionsbezogenen Projekt unter dem Namen mit dem Präfix `project_cvar_` und dem vom Benutzer festgelegten Variablennamen zur Verfügung.

3.3.8.5. Variablen für die einzelnen Zahlungseingänge

`payment`
: Betrag

`paymentaccount`
: Konto

`paymentdate`
: Datum

`paymentmemo`
: Memo

`paymentsource`
: Beleg

3.3.8.6. Benutzerdefinierte Kunden- und Lieferantenvariablen

Die vom Benutzer definierten Variablen für Kunden und Lieferanten stehen beim Ausdruck von Einkaufs- und Verkaufsbelegen ebenfalls zur Verfügung. Ihre Namen setzen sich aus dem Präfix `vc_cvar_` und dem vom Benutzer festgelegten Variablennamen zusammen.

Beispiel: Der Benutzer hat eine Variable namens `number_of_employees` definiert, die die Anzahl der Mitarbeiter des Unternehmens enthält. Diese Variable steht dann unter dem Namen `vc_cvar_number_of_employees` zur Verfügung.

Die benutzerdefinierten Variablen der Lieferadressen stehen unter einem ähnlichen Namensschema zur Verfügung. Hier lautet der Präfix `shiptocvar_`.

Analog stehen die benutzerdefinierten Variablen für Ansprechpersonen mit dem Namenspräfix `cp_cvar_` zur Verfügung.

Auch für das globale Projekt des Belegs stehen die benutzerdefinierten Variablen mit dem Namenspräfix `project_cvar_` zur Verfügung.

3.3.9. Variablen in Mahnungen und Rechnungen über Mahngebühren

3.3.9.1. Namen der Vorlagen

Die Namen der Vorlagen werden im System-Menü vom Benutzer eingegeben. Wird für ein Mahnlevel die Option zur automatischen Erstellung einer Rechnung über die Mahngebühren und Zinsen aktiviert, so wird der Name der Vorlage für diese Rech-

nung aus dem Vorlagenname für diese Mahnstufe mit dem Zusatz `_invoice` gebildet. Weiterhin werden die Kürzel für die ausgewählte Sprache und den ausgewählten Drucker angehängt.

3.3.9.2. Allgemeine Variablen in Mahnungen

Die Variablen des Bearbeiters, bzw. Verkäufers stehen wie gewohnt als `employee_...` bzw. `salesman_...` zur Verfügung. Werden mehrere Rechnungen in einer Mahnung zusammengefasst, so werden die Metadaten (Bearbeiter, Abteilung, etc) der ersten angemahnten Rechnung im Ausdruck genommen.

Die Adressdaten des Kunden stehen als Variablen `name`, `street`, `zipcode`, `city`, `country`, `department_1`, `department_2`, und `email` zur Verfügung. Der Ansprechpartner `cp_...` steht auch zu Verfügung, wird allerdings auch nur von der ersten angemahnten Rechnung (s.o.) genommen.

Weitere Variablen beinhalten:

`dunning_date`
: Datum der Mahnung

`dunning_duedate`
: Fälligkeitsdatum für diese Mahhnung

`dunning_id`
: Mahnungsnummer

`fee`
: Kumulative Mahngebühren

`interest_rate`
: Zinssatz per anno in Prozent

`total_amount`
: Gesamter noch zu zahlender Betrag als `fee` + `total_interest` + `total_open_amount`

`total_interest`
: Zinsen per anno über alle Rechnungen

`total_open_amount`
: Summe über alle offene Beträge der Rechnungen

3.3.9.3. Variablen für jede gemahnte Rechnung in einer Mahnung

`dn_amount`
: Rechnungssumme (brutto)

`dn_duedate`
: Originales Fälligkeitsdatum der Rechnung

`dn_dunning_date`
: Datum der Mahnung

`dn_dunning_duedate`
: Fälligkeitsdatum der Mahnung

`dn_fee`
: Kummulative Mahngebühr

`dn_interest`
: Zinsen per anno für diese Rechnung

`dn_invnumber`
: Rechnungsnummer

`dn_linetotal`
: Noch zu zahlender Betrag (ergibt sich aus `dn_open_amount` + `dn_fee` + `dn_interest`)

`dn_netamount`
: Rechnungssumme (netto)

`dn_open_amount`
: Offener Rechnungsbetrag

`dn_ordnumber`
: Bestellnummer

`dn_transdate`
: Rechnungsdatum

`dn_curr`
: Währung, in der die Rechnung erstellt wurde. (Die Rechnungsbeträge sind aber immer in der Hauptwährung)

3.3.9.4. Variablen in automatisch erzeugten Rechnungen über Mahngebühren

Die Variablen des Verkäufers stehen wie gewohnt als `employee_...` zur Verfügung. Die Adressdaten des Kunden stehen als Variablen `name`, `street`, `zipcode`, `city`, `country`, `department_1`, `department_2`, und `email` zur Verfügung.

Weitere Variablen beinhalten:

`duedate`
: Fälligkeitsdatum der Rechnung

`dunning_id`
: Mahnungsnummer

`fee`
: Mahngebühren

`interest`
: Zinsen

`invamount`
: Rechnungssumme (ergibt sich aus `fee` + `interest`)

`invdate`
: Rechnungsdatum

`invnumber`
: Rechnungsnummer

3.3.10. Variablen in anderen Vorlagen

3.3.10.1. Einführung

Die Variablen in anderen Vorlagen sind ähnlich wie in der Rechnung. Allerdings heißen die Variablen, die mit `inv` beginnen, jetzt anders. Bei den Angeboten fangen sie mit `quo` für "quotation" an: `quodate` für Angebotsdatum etc. Bei Bestellungen wiederum fangen sie mit `ord` für "order" an: `ordnumber` für Bestellnummer etc.

Manche Variablen sind in anderen Vorlagen hingegen gar nicht vorhanden wie z.B. die für bereits verbuchte Zahlungseingänge. Dies sind Variablen, die vom Geschäftsablauf her in der entsprechenden Vorlage keine Bedeutung haben oder noch nicht belegt sein können.

Im Folgenden werden nur wichtige Unterschiede zu den Variablen in Rechnungen aufgeführt.

3.3.10.2. Angebote und Preisanfragen

`quonumber`
: Angebots- bzw. Anfragenummer

`reqdate`
: Gültigkeitsdatum (bei Angeboten) bzw. Lieferdatum (bei Preisanfragen)

`transdate`
: Angebots- bzw. Anfragedatum

3.3.10.3. Auftragsbestätigungen und Lieferantenaufträge

`ordnumber`
: Auftragsnummer

`reqdate`
: Lieferdatum

`transdate`
: Auftragsdatum

3.3.10.4. Lieferscheine (Verkauf und Einkauf)

`cusordnumber`
: Bestellnummer des Kunden (im Verkauf) bzw. Bestellnummer des Lieferanten (im Einkauf)

`donumber`
: Lieferscheinnummer

`transdate`
: Lieferscheindatum

Für jede Position eines Lieferscheines gibt es ein Unterarray mit den Informationen darüber, von welchem Lager und Lagerplatz aus die Waren verschickt wurden (Verkaufslieferscheine) bzw. auf welchen Lagerplatz sie eingelagert wurden. Diese müssen mittels einer `foreach`-Schleife ausgegeben werden. Diese Variablen sind:

`si_bin`
: Lagerplatz

`si_chargenumber`
: Chargennummer

`si_bestbefore`
: Mindesthaltbarkeit

`si_number`
: Artikelnummer

`si_qty`
: Anzahl bzw. Menge

`si_nominal_qty`
: Der Soll-Bestand dieser Kombination aus Artikel, Charge, Mindesthaltbarkeit und Lagerplatz in kivitendo in der Einheit si_unit. Kann z.B. in der Sammelliste gedruckt werden, um beim Einlagern direkt Abweichungen erkennen zu können.

`si_runningnumber`
: Positionsnummer (1, 2, 3 etc)

`si_unit`
: Einheit

`si_warehouse`
: Lager

3.3.10.5. Variablen für Sammelrechnung

`c0total`
: Gesamtbetrag aller Rechnungen mit Fälligkeit < 30 Tage

`c30total`
: Gesamtbetrag aller Rechnungen mit Fälligkeit >= 30 und < 60 Tage

`c60total`
: Gesamtbetrag aller Rechnungen mit Fälligkeit >= 60 und < 90 Tage

`c90total`
: Gesamtbetrag aller Rechnungen mit Fälligkeit >= 90 Tage

`total`
: Gesamtbetrag aller Rechnungen

Variablen für jede Rechnungsposition in Sammelrechnung:

`invnumber`
: Rechnungsnummer

`invdate`
: Rechnungsdatum

`duedate`
: Fälligkeitsdatum

`amount`
: Summe der Rechnung

`open`
: Noch offener Betrag der Rechnung

`c0`
: Noch offener Rechnungsbetrag mit Fälligkeit < 30 Tage

`c30`
: Noch offener Rechnungsbetrag mit Fälligkeit >= 30 und < 60 Tage

`c60`
: Noch offener Rechnungsbetrag mit Fälligkeit >= 60 und < 90 Tage

`c90`
: Noch offener Rechnungsbetrag mit Fälligkeit >= 90 Tage

3.3.11. Blöcke, bedingte Anweisungen und Schleifen

3.3.11.1. Einführung

Der Parser kennt neben den Variablen einige weitere Konstrukte, die gesondert behandelt werden. Diese sind wie Variablennamen in spezieller Weise markiert: **<%anweisung%> ... <%end%>**

Anmerkung zum **<%end%>**: Der besseren Verständlichkeit halber kann man nach dem **end** noch beliebig weitere Wörter schreiben, um so zu markieren, welche Anweisung (z.B. **if** oder **foreach**) damit abgeschlossen wird.

Beispiel: Lautet der Beginn eines Blockes z.B. **<%if type == "sales_quotation"%>**, so könnte er mit **<%end%>** genauso abgeschlossen werden wie mit **<%end if%>** oder auch **<%end type == "sales_quotation"%>**.

3.3.11.2. Der if-Block

```
<%if variablenname%>
...
<%end%>
```

Eine normale "if-then"-Bedingung. Die Zeilen zwischen dem "if" und dem "end" werden nur ausgegeben, wenn die Variable `variablenname` gesetzt und ungleich 0 ist.

Handelt es sich bei der benannten Variable um ein Array, also um einen Variablennamen, über den man mit **<%foreach variablenname%>** iteriert, so wird mit diesem Konstrukt darauf getestet, ob das Array Elemente enthält. Somit würde im folgenden Beispiel nur dann eine Liste von Zahlungseingängen samt ihrer Überschrift "Zahlungseingänge" ausgegeben, wenn tatsächlich welche getätigt wurden:

```
<%if payment%>
Zahlungseingänge:
 <%foreach payment%>
   Am <%paymentdate%>: <%payment%> €
 <%end foreach%>
<%end if%>
```

Die Bedingung kann auch negiert werden, indem das Wort `not` nach dem `if` verwendet wird. Beispiel:

```
<%if not cp_greeting%>
...
<%end%>
```

Zusätzlich zu dem einfachen Test, ob eine Variable gesetzt ist oder nicht, bietet dieser Block auch die Möglichkeit, den Inhalt einer Variablen mit einer festen Zeichenkette oder einer anderen Variablen zu vergleichen. Ob der Vergleich mit einer Zeichenkette oder einer anderen Variablen vorgenommen wird, hängt davon ab, ob die rechte Seite des Vergleichsoperators in Anführungszeichen gesetzt wird (Vergleich mit Zeichenkette) oder nicht (Vergleich mit anderer Variablen). Zwei Beispiele, die beide Vergleiche zeigen:

```
<%if var1 == "Wert"%>
```

Testet die Variable `var1` auf übereinstimmung mit der Zeichenkette `Wert`. Mittels `!=` anstelle von `==` würde auf Ungleichheit getestet.

```
<%if var1 == var2%>
```

Testet die Variable `var1` auf übereinstimmung mit der Variablen `var2`. Mittel `!=` anstelle von `==` würde auf Ungleichheit getestet.

Erfahrere Benutzer können neben der Tests auf (Un-)Gleichheit auch Tests auf Übereinstimmung mit regulären Ausdrücken ohne Berücksichtung der Groß- und Kleinschreibung durchführen. Dazu dient dieselbe Syntax wie oben nur mit `=~` und `!~` als Vergleichsoperatoren.

Beispiel für einen Test, ob die Variable `intnotes` (interne Bemerkungen) das Wort `schwierig` enthält:

```
<%if intnotes =~ "schwierig"%>
```

3.3.11.3. Der foreach-Block

```
<%foreach variablenname%>
...
<%end%>
```

Fügt die Zeilen zwischen den beiden Anweisungen so oft ein, wie das Perl-Array der Variablen `variablenname` Elemente enthät. Dieses Konstrukt wird zur Ausgabe der einzelnen Posten einer Rechnung / eines Angebots sowie zur Ausgabe der Steuern benutzt. In jedem Durchlauf werden die zeilenbezogenen Variablen jeweils auf den Wert für die aktuelle Position gesetzt.

Die Syntax sieht normalerweise wie folgt aus:

```
<%foreach number%>
Position: <%runningnumber%>
Anzahl: <%qty%>
Artikelnummer: <%number%>
Beschreibung: <%description%>
...
<%end%>
```

Besonderheit in OpenDocument-Vorlagen: Tritt ein `<%foreach%>`-Block innerhalb einer Tabellenzelle auf, so wird die komplette Tabellenzeile so oft wiederholt wie notwendig. Tritt er außerhalb auf, so wird nur der Inhalt zwischen `<%foreach%>` und `<%end%>` wiederholt, nicht aber die komplette Zeile, in der er steht.

3.3.12. Markup-Code zur Textformatierung innerhalb von Formularen

Wenn der Benutzer innhalb von Formularen in kivitendo Text anders formatiert haben möchte, so ist dies begrenzt möglich. kivitendo unterstützt die Textformatierung mit HTML-ähnlichen Tags. Der Benutzer kann z.B. bei der Artikelbeschreibung auf einer Rechnung Teile des Texts zwischen Start- und Endtags setzen. Dieser Teil wird dann automatisch in Anweisungen für das ausgewählte Vorlagenformat (HTML oder PDF über LaTeX) umgesetzt.

Die unterstützen Formatierungen sind:

Text
: Text wird in Fettdruck gesetzt.

Text
: Text wird kursiv gesetzt.

<u>Text</u>
: Text wird unterstrichen.

<s>Text</s>
: Text wird durchgestrichen. Diese Formatierung ist nicht bei der Ausgabe als PDF über LaTeX verfügbar.

<bullet>
: Erzeugt einen ausgefüllten Kreis für Aufzählungen (siehe unten).

Der Befehl **<bullet>** funktioniert momentan auch nur in Latex-Vorlagen.

3.3.13. Hinweise zur Anrede

Das Flag "natürliche Person" (`natural_person`) aus den Kunden- oder Lieferantenstammdaten kann in den Druckvorlagen zusammen mit dem Feld "Anrede" (`greeting`) z.B. dafür verwendet werden, die Anrede zwischen einer allgemeinen und einer persönlichen Anrede zu unterscheiden.

```
<%if natural_person%><%greeting%> <%name%><%else%>Sehr geehrte Damen und Herren<%end if%>
```

3.4. Excel-Vorlagen

3.4.1. Zusammenfassung

Dieses Dokument beschreibt den Mechanismus, mit dem Exceltemplates abgearbeitet werden, und die Einschränkungen, die damit einhergehen.

3.4.2. Bedienung

Der Excel Mechanismus muss in der Konfigurationsdatei aktiviert werden. Die Konfigurationsoption heißt `excel_templates = 1` im Abschnitt `[print_templates]`.

Eine Excelvorlage kann dann unter dem Namen einer beliebigen anderen Vorlage mit der Endung `.xls` gespeichert werden. In den normalen Verkaufsmasken taucht nun `Excel` als auswählbares Format auf und kann von da an wie LaTeX- oder OpenOffice-Vorlagen benutzt werden.

Der Sonderfall der Angebote aus der Kundenmaske ist ebenfalls eine Angebotsvorlage und wird unter dem internen Namen der Angebote `sales_quotation.xls` gespeichert.

3.4.3. Variablensyntax

Einfache Syntax: **<<varname>>**

Dabei sind << und >> die Delimiter. Da Excel auf festen Breiten besteht, kann der Tag künstlich verlängert werden, indem weitere < oder > eingefügt werden. Der Tag muss nicht symmetrisch sein. Beispiel:

```
<<<<<varname>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Um die Limitierung der festen Breite zu reduzieren, können weitere Variablen in einem Block interpoliert werden. Whitespace wird dazwishen dann erhalten. Beispiel:

```
<<<<<varname1 varname2   varname3>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Die Variablen werden interpoliert, und linksbündig mit Leerzeichen auf die gewünschte Länge aufgefüllt. Ist der String zu lang, werden überzählige Zeichen abgeschnitten.

Es ist ausserdem möglich, Daten rechtsbündig darzustellen, wenn der Block mit einem Leerzeichen anfängt. Beispiel:

```
<<<<<<            varname>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Dies würde rechtsbündig triggern. Wenn bei rechtsbündiger Ausrichtung Text abgeschnitten werden muss, wird er vom linken Ende entfernt.

3.4.4. Einschränkungen

Das Excelformat bis 2002 ist ein binäres Format, und kann nicht mit vertretbarem Aufwand editiert werden. Der Templatemechanismus beschränkt sich daher darauf, Textstellen exakt durch einen anderen Text zu ersetzen.

Aus dem gleichen Grund sind die Kontrolllstrukturen **<%if%>** und **<%foreach%>** nicht vorhanden. Der Delimiter <% %> kommt in den Headerinformationen evtl. vor. Deshalb wurde auf den sichereren Delimiter << und >> gewechselt.

3.5. Mandantenkonfiguration Lager

Die Lagerverwaltung in kivitendo funktioniert standardmässig wie folgt: Wird ein Lager mit einem Lagerplatz angelegt, so gibt es die Möglichkeit hier über den Menüpunkt Lager entsprechende Warenbewegungen durchzuführen. Ferner kann jede Position eines Lieferscheins ein-, bzw. ausgelagert werden (Einkauf-, bzw. Verkauf). Es können beliebig viele Lager mit beliebig vielen Lagerplätzen abgebildet werden. Die Lagerbewegungen über einen Lieferschein erfolgt durch Anklicken jeder Einzelposition und das Auswählen dieser Position zu einem Lager mit Lagerplatz. Dieses Verfahren lässt sich schrittweise vereinfachen, je nachdem wie die Einstellungen in der Mandatenkonfiguration gesetzt werden.

- `Auslagern über Standardlagerplatz` Hier wird ein zusätzlicher Knopf (Auslagern über Standard-Lagerplatz) in dem Lieferschein-Beleg hinzugefügt, der dann alle Lagerbewegungen über den Standardlagerplatz (konfigurierbar pro Ware) durchführt.

- `Auslagern ohne Bestandsprüfung` Das obige Auslagern schlägt fehl, wenn die entsprechende Menge für die Lagerbewegung nicht vorhanden ist, möchte man dies auch ignorieren und ggf. dann nachpflegen, so kann man eine Negativ-Warenmenge mit dieser Option erlauben. Hierfür muss ein entsprechender Lagerplatz (Fehlbestand, o.ä.) konfiguriert sein.

Zusätzliche Funktionshinweise:

- `Standard-Lagerplatz` Ist dieser konfiguriert, wird dies auch als Standard-Voreinstellung bei der Neuerfassung von Stammdaten → Waren / Dienstleistung / Erzeugnis verwendet.
- `Standard-Lagerplatz verwenden, falls keiner in Stammdaten definiert` Wird beim 'Auslagern über Standardlagerplatz' keine Standardlagerplatz zu der Ware gefunden, so wird mit dieser Option einfach der Standardlagerplatz verwendet.

3.6. Schweizer Kontenpläne

Seit der Version 3.5 stehen in kivitendo 3 Kontenpläne für den Einsatz in der Schweiz zur Verfügung, einer für Firmen und Organisationen, die nicht mehrwertsteuerpflichtig sind, einer für Firmen, die mehrwertsteuerpflichtig sind und einer speziell für Vereine.

Die Kontenpläne orientieren sich am in der Schweiz üblicherweise verwendeten KMU-Kontenrahmen und sind mit der Revision des Schweizerischen Obligationenrechts (OR) vom 1.1.2013 kompatibel, insbesondere `Art.957a Abs.2`.

Beim Vereinskontenplan sind standardmässig nur die Konten 1100 (Debitoren CHF) und 1101 (Debitoren EUR) als Buchungskonten im Verkauf sowie die Konten 2000 (Kreditoren CHF) und 2001 (Kreditoren EUR) als Buchungskonten im Einkauf vorgesehen. Weitere Konten können bei Bedarf in den Konto-Detaileinstellungen als Einkaufs- oder Verkaufskonten konfiguriert werden.

Die Möglichkeit, Saldosteuersätze zu verwenden ist in der aktuellen Version von kivitendo noch nicht integriert.

Trotzdem können auch Firmen, die per Saldosteuersatz mit der Eidgenössischen Steuerverwaltung abrechnen, kivitendo bereits nutzen. Dazu wird der Kontenplan mit MWST ausgewählt. Anschliessend müssen alle Aufwandskonten editiert werden und dort der Steuersatz auf 0% gesetzt werden.

So werden bei Kreditorenbuchungen keine Vorsteuern verbucht.

Bezugssteuern für aus dem Ausland bezogene Dienstleistungen müssen manuell verbucht werden.

Wünsche für Anpassungen an den Schweizer Kontenplänen sowie Vorschläge für weitere (z.B. branchenspezifische) Kontenpläne bitte an `empfang@revamp-it.ch` senden.

3.7. Artikelklassifizierung

3.7.1. Übersicht

Die Klassifizierung von Artikeln dient einer weiteren Gliederung, um zum Beispiel den Einkauf vom Verkauf zu trennen, gekennzeichnet durch eine Beschreibung (z.B. "Einkauf") und ein Kürzel (z.B. "E"). Für jede Klassifizierung besteht eine Beschreibung und eine Abkürzung die normalerweise aus einem Zeichen besteht, kann aber auf mehrere Zeichen erweitert werden, falls zur Unterscheidung notwendig. Sinnvoll sind jedoch nur maximal 2 Zeichen.

3.7.2. Basisklassifizierung

Als Basisklassifizierungen gibt es

- Einkauf

- Verkauf
- Handelsware
- Produktion
- - keine - (diese wird bei einer Aktualisierung für alle existierenden Artikel verwendet und ist gültig für Verkauf und Einkauf)

Es können weitere Klassifizierungen angelegt werden. So kann es z.B. für separat auszuweisende Artikel folgende Klassen geben:

- Lieferung (Logistik, Transport) mit Kürzel L
- Material (Verpackungsmaterial) mit Kürzel M

3.7.3. Attribute

Bisher haben die Klassifizierungen folgende Attribute, die auch alle gleichzeitg gültig sein können

- gültig für Verkauf - dieser Artikel kann im Verkauf genutzt werden
- gültig für Einkauf - dieser Artikel kann im Einkauf genutzt werden
- separat ausweisen - hierzu gibt es zur Dokumentengenerierung (LaTeX) eine zusätzliche Variable

Für das Attribut "separat ausweisen" stehen in den LaTeX-Vorlagen die Variable **<%non_separate_subtotal%>** zur Verfügung, die alle nicht separat auszuweisenden Artikelkosten saldiert, sowie pro separat auszuweisenden Klassifizierungen die Variable**< %separate_X_subtotal%>**, wobei X das Kürzel der Klassifizierung ist.

Im obigen Beispiel wäre das für Lieferkosten **<%separate_L_subtotal%>** und für Verpackungsmaterial **< %separate_M_subtotal%>**.

3.7.4. Zwei-Zeichen Abkürzung

Der Typ des Artikels und die Klassifizierung werden durch zwei Buchstaben dargestellt. Der erste Buchstabe ist eine Lokalisierung des Artikel-Typs ('P','A','S'), deutsch 'W', 'E', und 'D' für Ware Erzeugnis oder Dienstleistung und ggf. weiterer Typen.

Der zweite Buchstabe (und ggf. auch ein dritter, falls nötig) entspricht der lokalisierten Abkürzung der Klassifizierung.

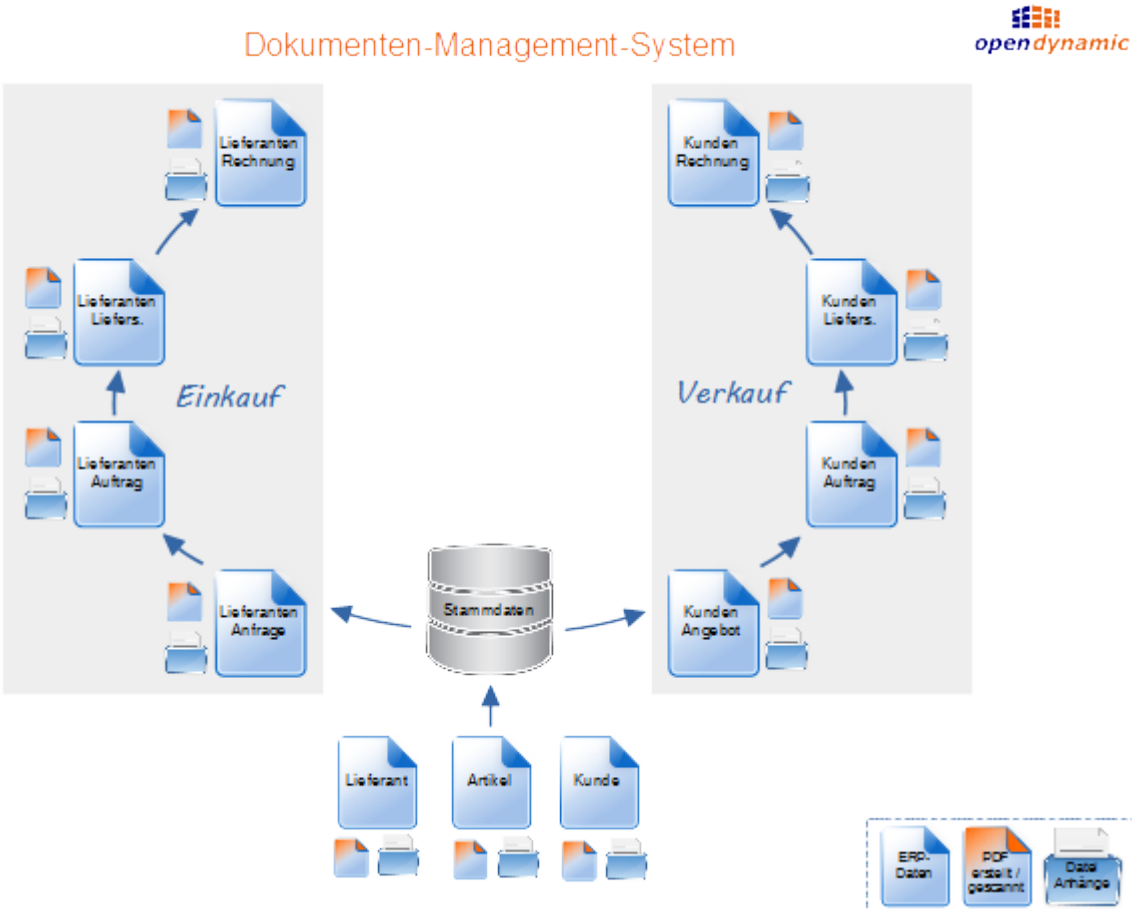
Diese Abkürzung wird überall beim Auflisten von Artikeln zur Erleichterung mit dargestellt.

3.8. Dateiverwaltung (Mini-DMS)

3.8.1. Übersicht

Parallel zum alten WebDAV gibt es ein Datei-Management-System, das Dateien verschiedenen Typs verwaltet. Dies können

1. aus ERP-Daten per LaTeX Template erzeugte PDF-Dokumente,
2. zu bestimmten ERP-Daten gehörende Anhangdateien unterschiedlichen Formats,
3. per Scanner eingelesene PDF-Dateien,
4. per E-Mail empfangene Dateianhänge unterschiedlichen Formats,
5. sowie speziel für Artikel hochgeladene Bilder sein.



3.8.2. Struktur

Über eine vom Speichermedium unabhängige Zwischenschicht werden die Dateien und ihre Versionen in der Datenbank verwaltet. Darunter können verschiedene Implementierungen (Backends) gleichzeitig existieren:

- Dateisystem
- WebDAV
- Schnittstelle zu externen Dokumenten-Management-Systemen
- andere Datenbank
- etc ...

Es gibt unterschiedliche Typen von Dateien. Jedem Typ läßt sich in der Mandantenkonfiguration ein bestimmtes Backend zuordnen.

- "document": Das sind entweder generierte, eingescannte oder hochgeladene PDF-Dateien, die zu bestimmten ERP-Daten (ERP-Objekte, wie z.B. Rechnung, Lieferschein) gehören.
- "attachment": zusätzlich hochgeladene Dokumente, die an bestimmte ERP-Objekte angehängt werden, z.B. technische Zeichnungen, Aufmaße. Diese können auch für Artikel, Lieferanten und Kunden hinterlegt sein.
- "image": Bilder für Artikel. Diese können auch verkleinert in einer Vorschau (Thumbnail) angezeigt werden.

Zusätzlich werden in der Datenbank zu den Dateien neben der Zuordnung zu ERP-Objekten, Dateityp Dateinamen und Backend, in dem die Datei gespeichert ist, auch die Quelle der Datei notiert:

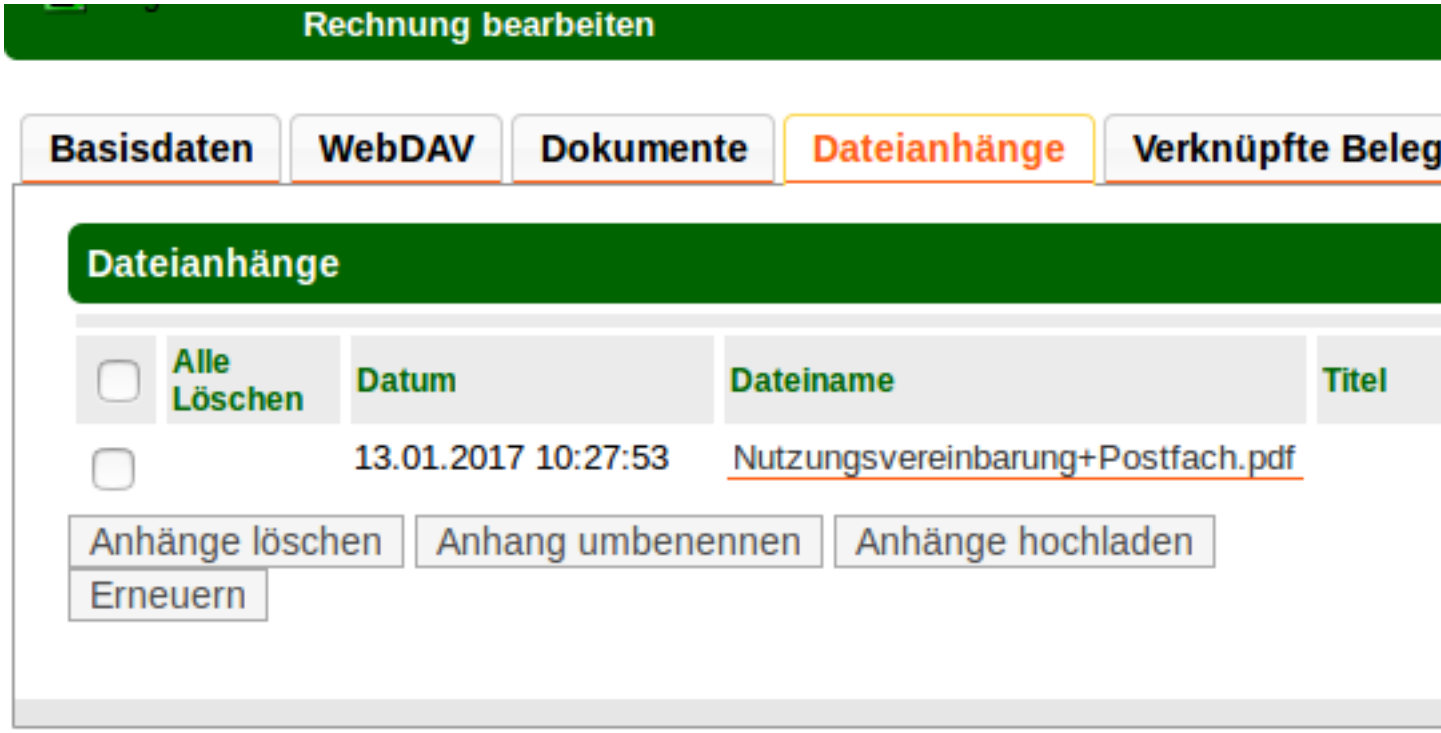
- "created": vom System erzeugte Dokumente"
- "uploaded": hochgeladene Dokumente

- "email": vom Mail-System empfangene Dateien
- "scanner[1]": von einem oder mehreren Scannern erzeugte Dateien. Existieren mehrere Scanner, so sind diese durch unterschiedliche Quellennamen zu definieren.

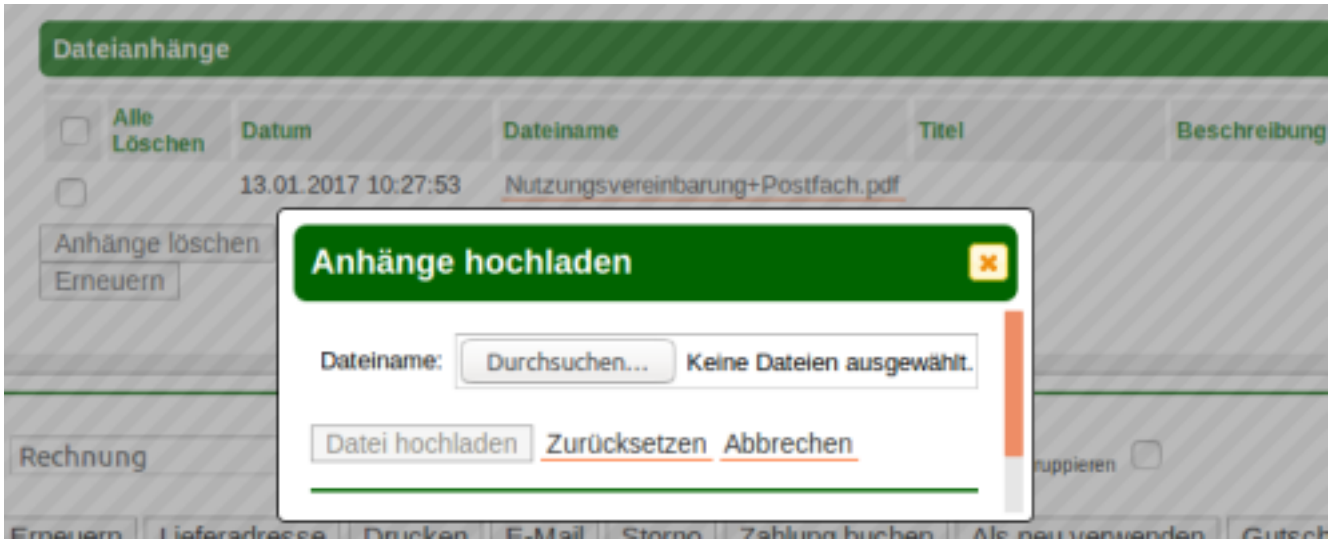
Je nach Dateityp sind nur bestimmte Quellen zulässig. So gibt es für "attachment" und "image" nur die Quelle "uploaded". Für "document" gibt es auf jeden Fall die Quelle "created". Die Quellen "scanner" und "email" müssen derzeit in der Datenbank konfiguriert werden (siehe Datenbank-Konfigurierung [84]).

3.8.3. Anwendung

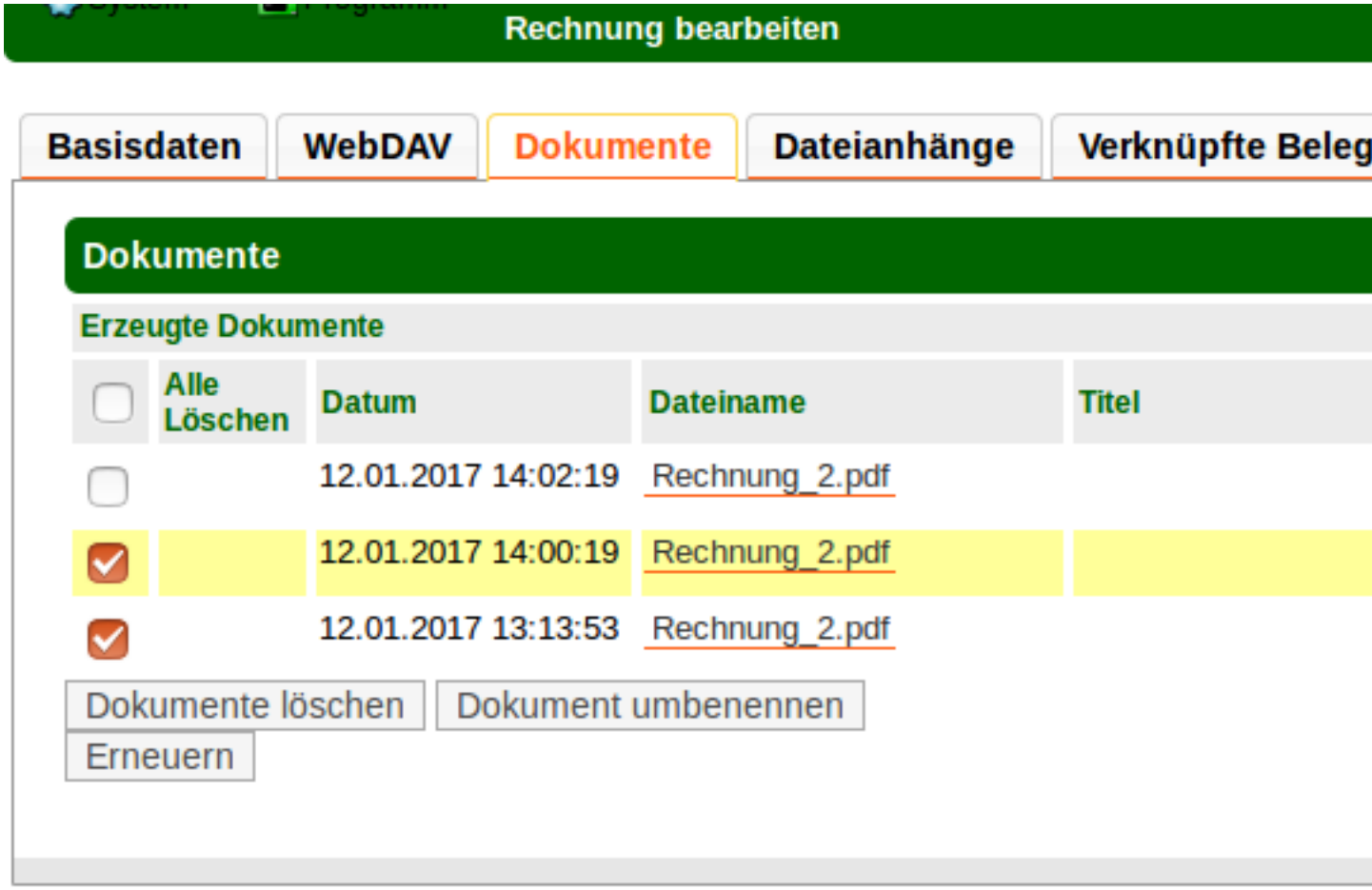
Die Daten werden bei den ERP-Objekten als extra Reiter dargestellt. Eine Verkaufsrechnung z.B. hat die Reiter "Dokumente" und "Dateianhänge".



Bei den Dateianhängen wird immer nur die aktuelle Version einer Datei angezeigt. Wird eine Datei mit gleichem Namen hochgeladen, so wird eine neue Version der Datei erstellt. Vorher wird der Anwender durch einen Dialog gefragt, ob er eine neue Version anlegen will oder ob er die Datei umbenennen will, falls es eine neue Datei sein soll.



Es können mehrere Dateien gleichzeitig hochgeladen werden, solange in Summe die maximale Größe nicht überschritten wird (siehe Mandantenkonfigurierung [82]).



Sind keine weiteren Quellen für Dokumente konfiguriert, so gibt es nur "erzeugte Dokumente". Es werden alle Versionen der generierten Datei angezeigt. Für Verkaufsrechnungen kommen keine anderen Quellen zur Geltung. Werden entsprechend der Datenbank-Konfigurierung [84] zusätzliche Quellen konfiguriert, so sind diese z.B. bei Einkaufsrechnungen sichtbar:

Statt des Löschens wird hier die Datei zurück zur Quelle verschoben. Somit kann die Datei anschließend an ein anderes ERP-Objekt angehängt werden.

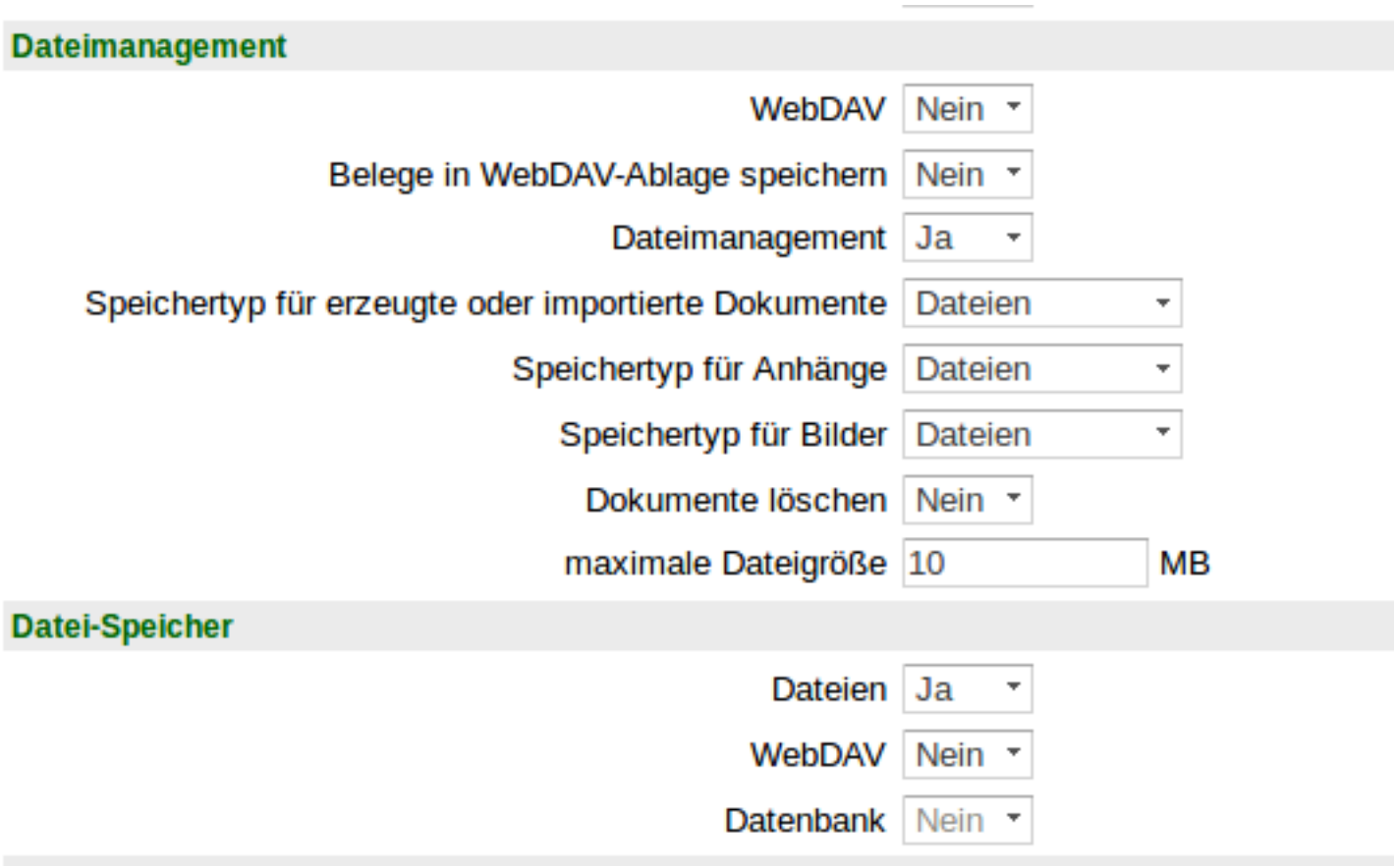
Derzeit sind "Titel" und "Beschreibung" noch nicht genutzt. Sie sind bisher nur bei Bildern relevant.

3.8.4. Konfigurierung

3.8.4.1. Mandantenkonfiguration

3.8.4.1.1. Reiter "Features"

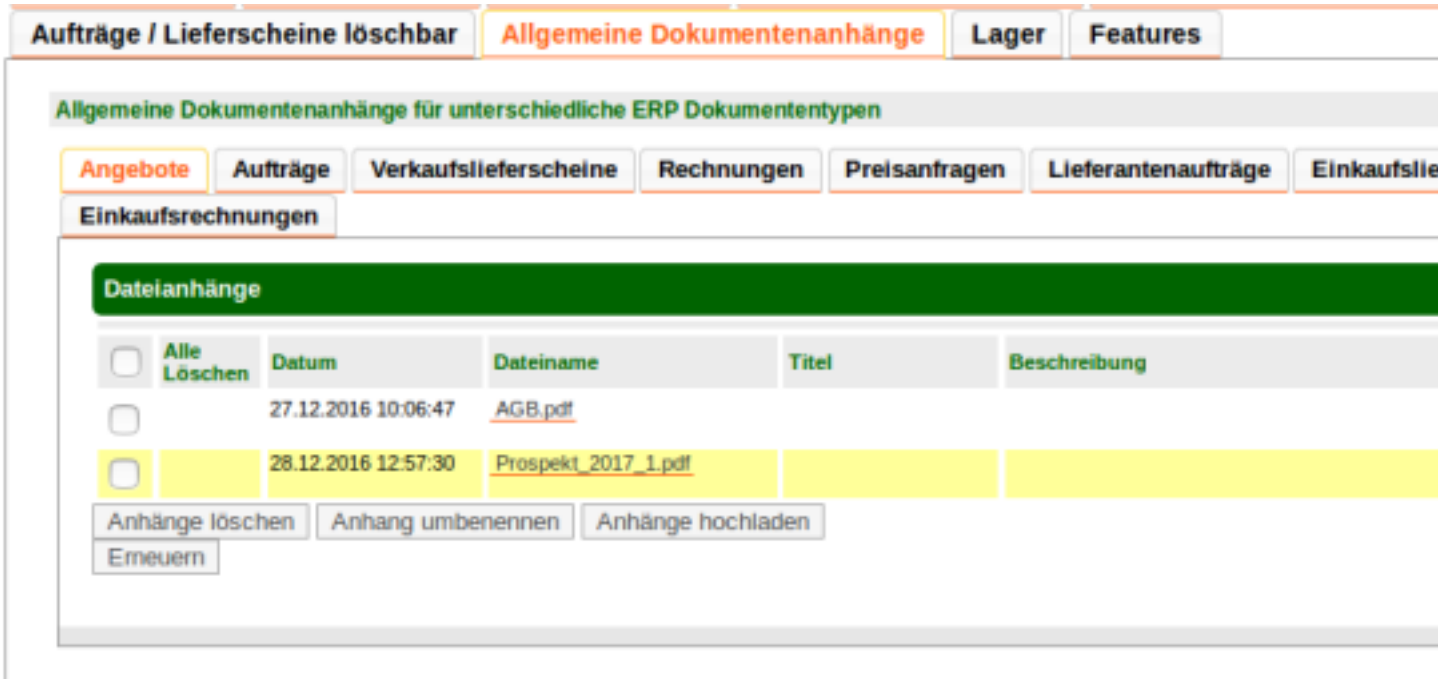
Unter dem Reiter **Features** im Abschnitt Dateimanagement ist neben dem "alten" WebDAV das Dateimangement generell zu- und abschaltbar, sowie die Zuordnung der Dateitypen zu Backends. Die Löschbarkeit von Dateien, sowie die maximale Uploadgröße sind Backend-unabhängig



Die einzelnen Backends sind einzeln einschaltbar. Spezifische Backend-Konfigurierungen sind hier noch ergänzbar.

3.8.4.1.2. Reiter "Allgemeine Dokumentenanhänge"

Unter dem Reiter **Allgemeine Dokumentenanhänge** kann für alle ERP-Dokumente (Angebote, Aufträge, Lieferscheine, Rechnungen im Verkauf und Einkauf) allgemeingültige Anhänge hochgeladen werden.



Diese Anhänge werden beim Generieren von PDF-Dateien an die ERP-Dokumente angehängt, z.B. AGBs oder aktuelle Angebote. Es werden in dem Fall die Daten kopiert, sodass an den ERP-Dokumenten immer die Anhänge zum Generierungszeitpunkt eingebettet sind.

3.8.4.2. Datenbank-Konfigurierung

Die zusätzlichen Quellen für "email" oder ein oder mehrere Scanner sind derzeit vom Administrator direkt in der Datenbanktabelle "user_preferences" einzurichten. Die "value" ist im JSON-Format mit den jeweiligen Werten des Verzeichnisses und der Beschreibung der Quelle.

```
 id |  login    |  namespace   | version |   key    |          value
----+-----------+--------------+---------+----------+--------------------------
  1 | #default# | file_sources | 0.00000 | scanner1 |
                              {"dir":"/var/tmp/scanner1","desc":"Scanner Einkauf"}
  2 | #default# | file_sources | 0.00000 | scanner2 |
                              {"dir":"/var/tmp/scanner2","desc":"Scanner Verkauf"}
  3 | #default# | file_sources | 0.00000 | emails   |
                              {"dir":"/var/tmp/emails","desc":"Empfangene Mails" }
```

Es ist daran gedacht, statt dem Default-Eintrag später für bestimmte Benutzer ('login') bestimmte Quellen zuzulassen. Dies wird nach Bedarf implementiert.

3.8.4.3. kivitendo-Konfigurationsdatei

Dort ist im Abschnitt [paths] der relative oder absolute Pfad zum Dokumentenwurzelverzeichnis einzutragen. Dieser muss für den Webserver schreib- und lesbar sein, jedoch nicht ausführbar.

```
[paths]
document_path = /var/local/kivi_documents
```

Unter diesem Wurzelverzeichnis wird pro Mandant automatisch ein Unterverzeichnis mit der ID des Mandanten angelegt.

3.9. Webshop-Api

Das Shopmodul bietet die Möglichkeit Onlineshopartikel und Onlineshopbestellungen zu verwalten und zu bearbeiten.

Es ist Multishopfähig, d.h. Artikel können mehreren oder unterschiedlichen Shops zugeordnet werden. Bestellungen können aus mehreren Shops geholt werden.

Zur Zeit bietet das Modul nur einen Connector zur REST-Api von Shopware. Weitere Connectoren können dazu programmiert und eingerichtet werden.

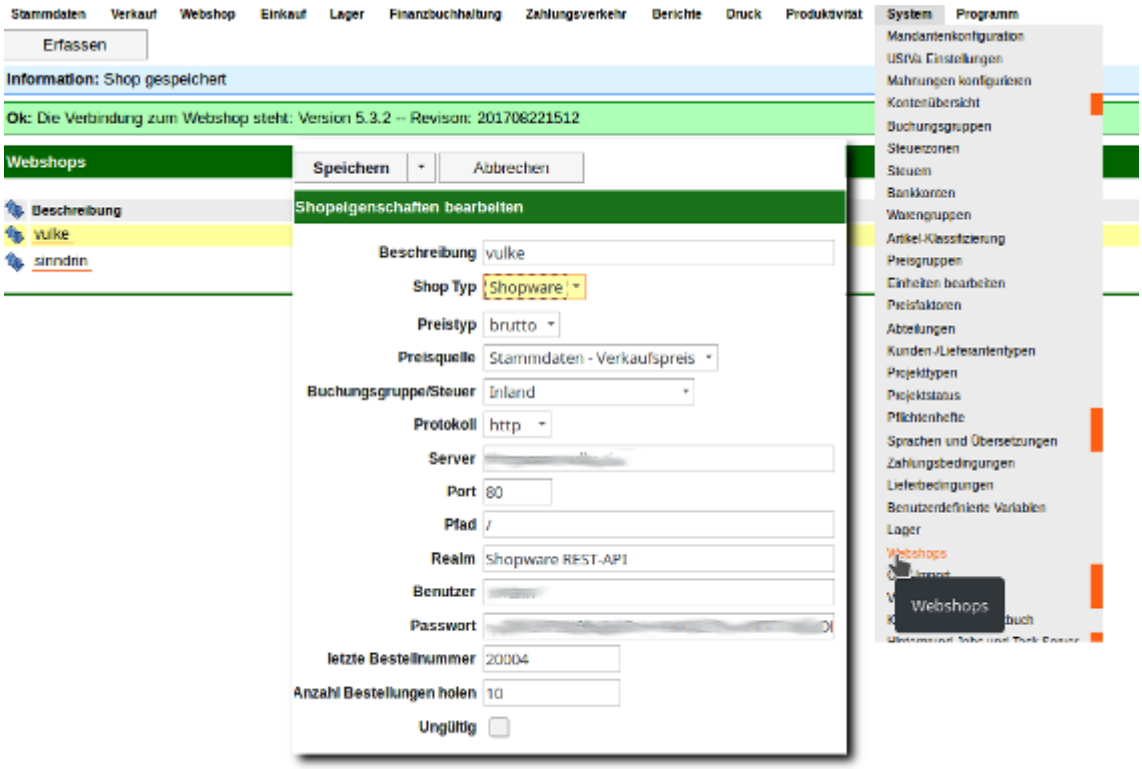
3.9.1. Rechte für die Webshopapi

In der Administration können folgende Rechte vergeben werden

- Webshopartikel anlegen und bearbeiten
- Shopbestellungen holen und bearbeiten
- Shop anlegen und bearbeiten

3.9.2. Konfiguration

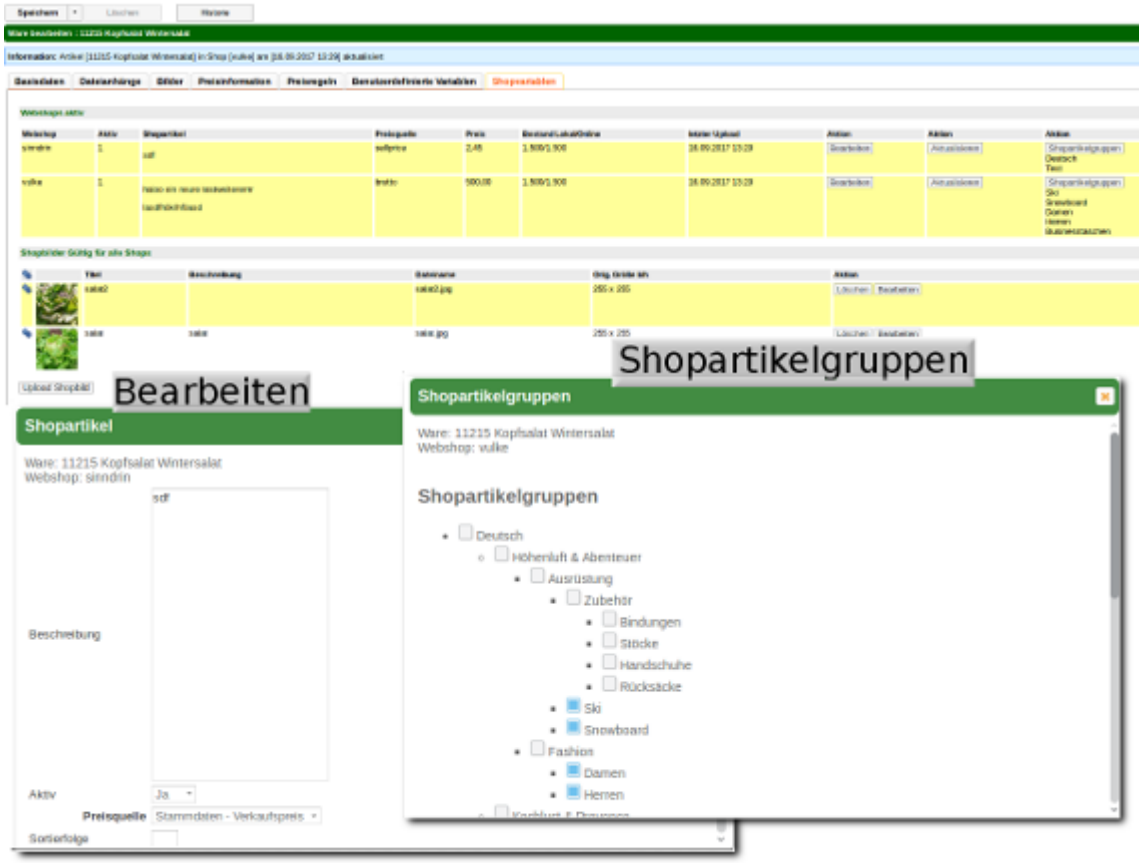
Unter System->Webshops können Shops angelegt und konfiguriert werden



3.9.3. Webshopartikel

3.9.3.1. Shopvariablenreiter in Artikelstammdaten

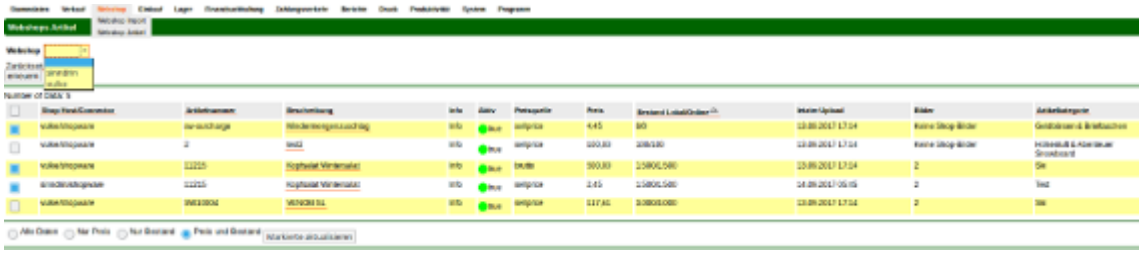
Mit dem Recht "Shopartikel anlegen und bearbeiten" und des Markers **"Shopartikel" in den Basisdaten** zeigt sich der Reiter "Shopvariablen" in den Artikelstammdaten. Hier können jetzt die Artikel mit unterschiedlichen Beschreibung und/oder Preisen für die konfigutierten Shops angelegt und bearbeitet werden. An dieser Stelle können auch beliebig viele Bilder dem Shopartikel zugeordnet werden. Artikelbilder gelten für alle Shops.



Die Artikelgruppen werden direkt vom Shopsystem geholt somit ist es möglich einen Artikel auch mehreren Gruppen zuzuordenen

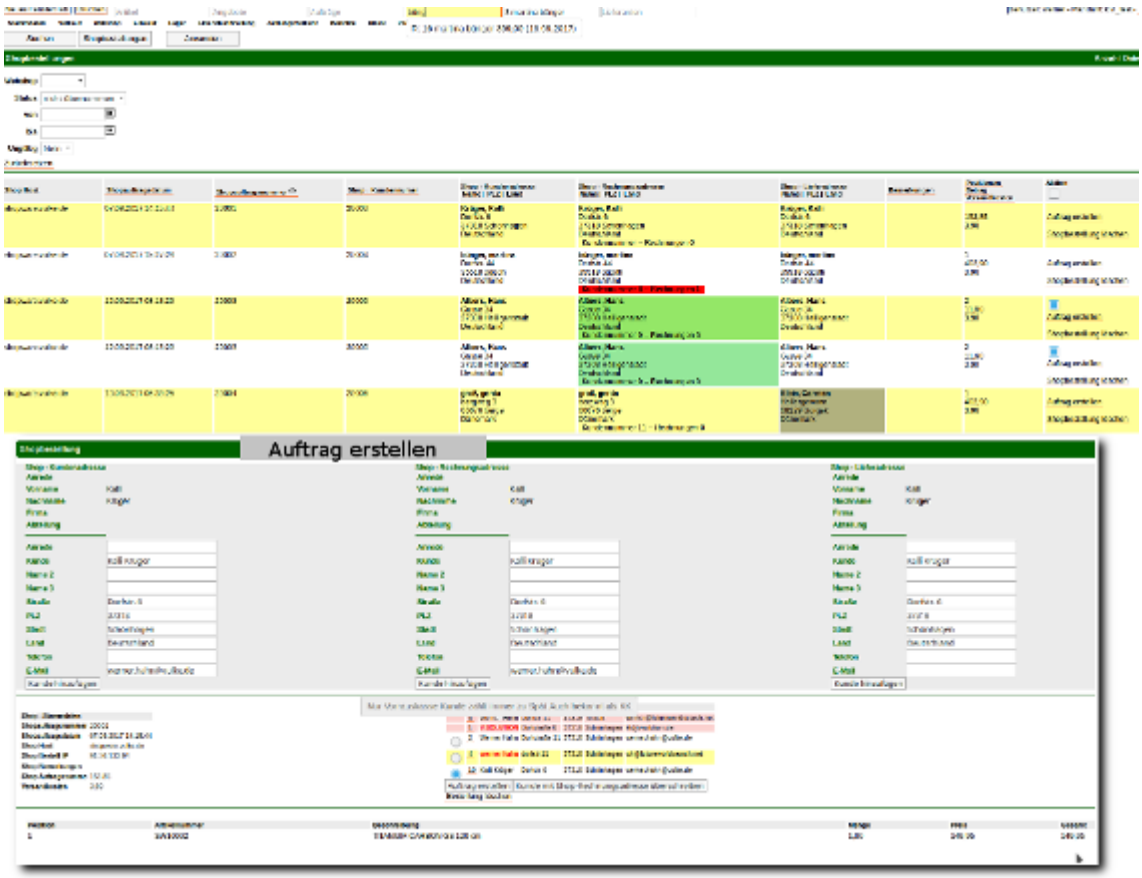
3.9.3.2. Shopartikelliste

Unter dem Menu Webshop->Webshop Artikel hat man nochmal eine Gesamtübersicht. Von hier aus ist es möglich Artikel im Stapel unter verschiedenen Kriterien <alles><nur Preis><nur Bestand><Preis und Bestand> an die jeweiligen Shops hochzuladen.



3.9.4. Bestellimport

Unter dem Menupunkt Webshop->Webshop Import öffnet sich die Bestellimportsliste. Hier ist sind Möglichkeiten gegeben Neue Bestellungen vom Shop abzuholen, geholte Bestellungen im Stapel oder einzeln als Auftrag zu transferieren. Die Liste kann nach verschiedenen Kriterien gefiltert werden.



Bei Einträgen in der Liste.

- keine Kundennummer: Es gibt ähnliche Kundendatensätze und der Datensatz konnte nicht eindeutig zugewiesen werden.
- Kundennummer und Rechnungen rot hinterlegt: Der Kunde hat offene Posten und kann deswegen nicht im Stapel übernommen werden.
- Rechnungsadresse grün hinterlegt: Der Kunde konnte eindeutig einem Datensatz zugeordnet werden. Die Shopbestellung kann im Stapel mit dem Button "Anwenden" und wenn markiert als Auftrag übernommen werden.
- Kundennummer vorhanden, aber die Checkbox "Auftrag erstellen" fehlt. Der Kunde hat vermutlich eine Shopauftragssperre.
- Lieferadresse grau hinterlegt: Optische Anzeige, dass es sich um eine unterschiedliche Lieferadresse handelt. Lieferadressen werden aber grundsätzlich beim Transferieren zu Aufträgen mit übernommen.
- In der Spalte Positionen/Betrag/Versandkosten zeigt sich ein tooltip zu den Positionen.

Maske Auftrag erstellen

Viele Shopsysteme haben drei verschieden Adresstypen Kunden-, Rechnungs-, und Lieferadresse, die sich auch alle unterscheiden können. Diese werden im oberen Bereich angezeigt. Es ist möglich jede dieser Adresse einzeln in kivitendo als Kunde zu übernehmen. Es werden die Werte Formulareingabe übernommen. Es wird bei einer Änderung allerdings nur diese in die kivitendo Kundenstammdaten übernommen, die Shopbestellung bleibt bestehen.

Mit der mittleren Adresse(Rechnungsadresse) im oberen Bereich, kann ich den ausgewählten kivitendodatensatz des mittleren Bereich überschreiben. Das ist sinnvoll, wenn ich erkenne, das der Kunde z.B. umgezogen ist.

Im mittleren Bereich das Adresslisting zeigt:

- Rot hinterlegt: Kunde hat eine Shopauftragssperre, diese muss zuerst deaktiviert werden bevor ich diesem Kunden eine Shopbestellung zuordnen kann.

- Kundenname fett und rot: Hier hat der Kunde eine Bemerkung in den Stammdaten. Ein Tooltip zeigt diese Bemerkung. Das kann dan auch der Grund für die Auftragssperre sein.
- Die Buttons "Auftrag erstellen" und "Kunde mit Rechnungsadresse überschreiben" zeigen sich erst, wenn ein Kunde aus dem Listing ausgewählt ist.
- Es ist aber möglich die Shopbestellung zu löschen.
- Ist eine Bestellung schon übernommen, zeigen sich an dieser Stelle, die dazugehörigen Belegverknüpfungen.

3.9.5. Mapping der Daten

Das Mapping der kivitendo Daten mit den Shopdaten geschieht in der Datei SL/ShopConnector/<SHOPCONNECTORNAME>.pm z.B.:SL/ShopConnector/Shopware.pm

In dieser Datei gibt es einen Bereich wo die Bestellpostionen, die Bestellkopfdaten und die Artikeldaten gemapt werden. In dieser Datei kann ein individelles Mapping dann gemacht werden. Zu Shopware gibt es hier eine sehr gute Dokumentation: https://developers.shopware.com/developers-guide/rest-api/

3.10. ZUGFeRD Rechnungen

3.10.1. Vorbedingung

Für die Erstellung von ZUGFeRD PDFs wird TexLive2018 oder höher benötigt.

Anmerkung

Wer kein TexLive2018 oder höher installieren kann, kann eine lokale Umgebung nur für kivitendo wie folgt erzeugen:

```
1. Download des offiziellen Installers von https://www.tug.org/texlive/

2. Installer ausführen, Standard-Ort für Installation belassen, evtl. e

3. Ein kleine Script »run_pdflatex.sh« anlegen, das den PATH auf das  I

------------------------------------------------------------
#!/bin/bash

export PATH=/usr/local/texlive/2020/bin/x86_64-linux:$PATH
hash -r

exec latexmk --pdflatex "$@"
------------------------------------------------------------

4. In config/kivitendo.conf den Parameter »latex« auf den Pfad zu »run_

5. Webserver neu starten
```

3.10.2. Übersicht

Mit der Version 3.5.6 bietet kivitendo die Möglichkeit ZUGFeRD Rechnungen zu erstellen, sowie auch ZUGFeRD Rechnungen direkt in kivitendo einzulesen.

Bei ZUGFeRD Rechnungen handelt es sich um eine PDF Datei in der eine XML-Datei eingebettet ist. Der Aufbau der XML-Datei ist standardisiert und ermöglicht so den Austausch zwischen den verschiedenen Softwareprodukten. Kivitendo setzt mit der Version 3.5.6 den ZUGFeRD 2.1 Standard um.

Weiter Details zu ZUGFeRD sind unter diesem Link zu finden: https://www.ferd-net.de/standards/was-ist-zugferd/index.html

3.10.3. Erstellen von ZUGFeRD Rechnungen in Kivitendo

Für die Erstellung von ZUGFeRD Rechnungen bedarf es in kivitendo zwei Dinge:

1. Die Erstellung muss in der Mandantenkonfiguration aktiviert sein
2. Beim mindestens einem Bankkonto muss die Option „Nutzung von ZUGFeRD“ aktiviert sein

3.10.3.1. Mandantenkonfiguration

Die Einstellung für die Erstellung von ZUGFeRD Rechnungen erfolgt unter „System“ → „Mandatenkonfiguration“ → „Features“. Im Abschnitt „Einkauf und Verkauf“ finden Sie die Einstellung „Verkaufsrechnungen mit ZUGFeRD-Daten erzeugen“. Hier besteht die Auswahl zwischen:

- ZUGFeRD-Rechnungen erzeugen
- ZUGFeRD-Rechnungen im Testmodus erzeugen
- Keine ZUGFeRD Rechnungen erzeugen

Rechnungen die als PDF erzeugt werden, werden je nach Einstellung nun im ZUGFeRD Format ausgegeben.

3.10.3.2. Konfiguration der Bankkonten

Unter „System → Bankkonten“ muss bei mindestens einem Bankkonto die Option „Nutzung mit ZUGFeRD“ auf „Ja“ gestellt werden.

3.10.4. Einlesen von ZUGFeRD Rechnungen in Kivitendo

Es lassen sich auch Rechnungen von Kreditoren, die im ZUGFeRD Format erstellt wurden, nach Kivitendo importieren. Hierfür müssen auch zwei Voraussetzungen erfüllt werden:

1. Beim Lieferanten muss die Umsatzsteuer-ID und das Bankkonto hinterlegt sein
2. Für den Kreditoren muss eine Buchungsvorlage existieren.

Wenn diese Voraussetzungen erfüllt sind, kann die Rechnung über „Finanzbuchhaltung“ → „Factur-X-/ZUGFeRD-Import“ über die „Durchsuchen“ Schaltfläche ausgewählt werden und über die Schaltfläche „Import“ eingeladen werden. Es öffnet sich daraufhin die Kreditorenbuchung. Die auslesbaren Daten aus dem eingebetteten XML der PDF Datei werden in der Kreditorenbuchung ergänzt.

3.11. Reklamationen

Reklamationen dienen dazu im Verkauf die Kundenzufriedenheit zu verbessern und Fehler in der Abwicklung von Aufträgen aufzuspüren und dann zu vermeiden.

Im Einkauf helfen Reklamationen beanstandete Lieferungen zu verwalten und zu kontrollieren.

3.11.1. Konfiguration des Reklamationsmodul

Unter dem Menu "System->Reklamationsgründe" muss mindestens ein Reklamationsgrund erfasst werden.

Stammdaten Verkauf Webshop Einkauf Lager

Anlegen Abbrechen

Reklamationsgrund hinzufügen

Name Fehlende Ware

Beschreibung Fehlende Ware

Gültig für Verkauf ☑

Gültig für Einkauf ☑

Aufgrund der Gründe können später in den Berichten Auswertungen gemacht werden.

Erfassen Sortieren

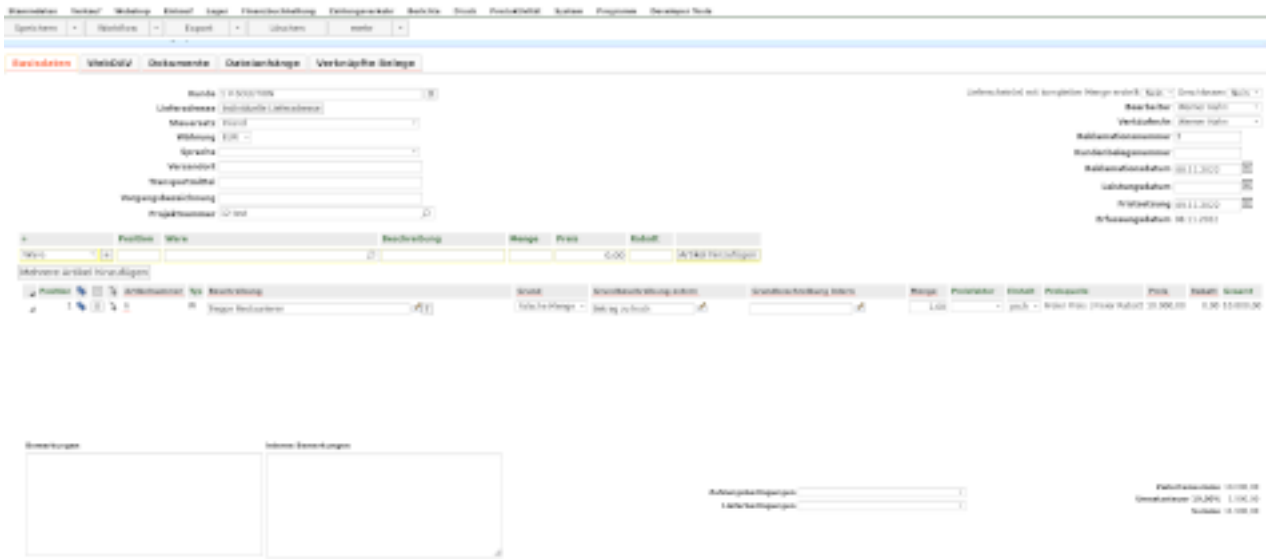
Information: Der Reklamationsgrund wurde erstellt.

Reklamationsgründe

Reklamationsgrund	Beschreibung	Gültig für Verkauf	Gültig für Einkauf
Falsche Menge	Falsche Menge	Ja	Ja
Falsche Maße	Falsches Maß	Ja	Ja
Falsche Ware	Falsche Ware	Ja	Ja
Fehlende Ware	Fehlende Ware	Ja	Ja

3.11.2. Reklamation erfassen

Aus den Belegen EK/VK Auftrag, EK/VK Lieferschein, EK/VK Rechnung kann unter "Workflow->Reklöamation erfassen" eine dem Workflow zugehörige Reklamation erfasst werden. Die Reklamationsmaske ähnelt im großem die der Auftragsmaske.



Es werden alle Positionen aus dem Beleg übernomen. Deswegen müssen hie die Positionen angepasst werden. Zusätzlich muss noch ein Reklamationsgrund angegeben werden.

Die Reklamationsgründe sind für spätere Auswertungen notwendig

Anmerkung

Anmerkung: Da bei einem Auftrag, sowohl im EK als auch im VK, noch keine Ware versendet bzw. empfangen wurde, könnte hier ein Reklamationsgrund "Lieferver sein.

3.11.3. Reklamationen auswerten

Bisher gibt es nur eine Liste mit Reklamationen und deren Filtermöglichkeiten.

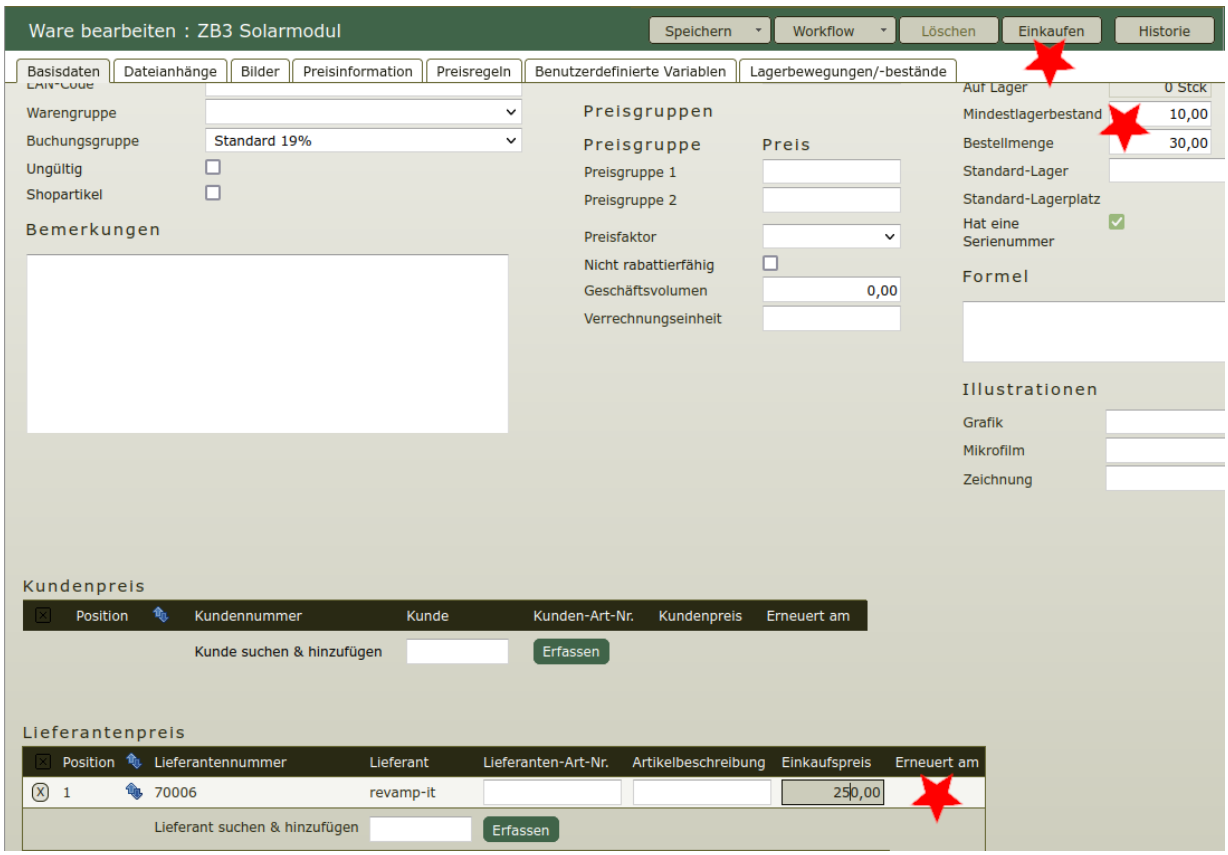
Einen individuellen Reklamationsbericht kann über einen "benutzerdefinierten Datenexport" erstellt werden

3.12. Dispositionsmanager/Einkaufshelfer

Der Einkaufshelfer generiert Lieferantenaufträge anhand eines Warenkorbes. Der Warenkorb kann einzeln oder mit dem Meldebestandsbericht gefüllt werden. Mit dem Filter „Lieferant“ im Warenkorb werden die Artikel einem Lieferanten zugeordnet und können dann bestellt, d.h. ein Lieferantenauftragerstellt, werden.

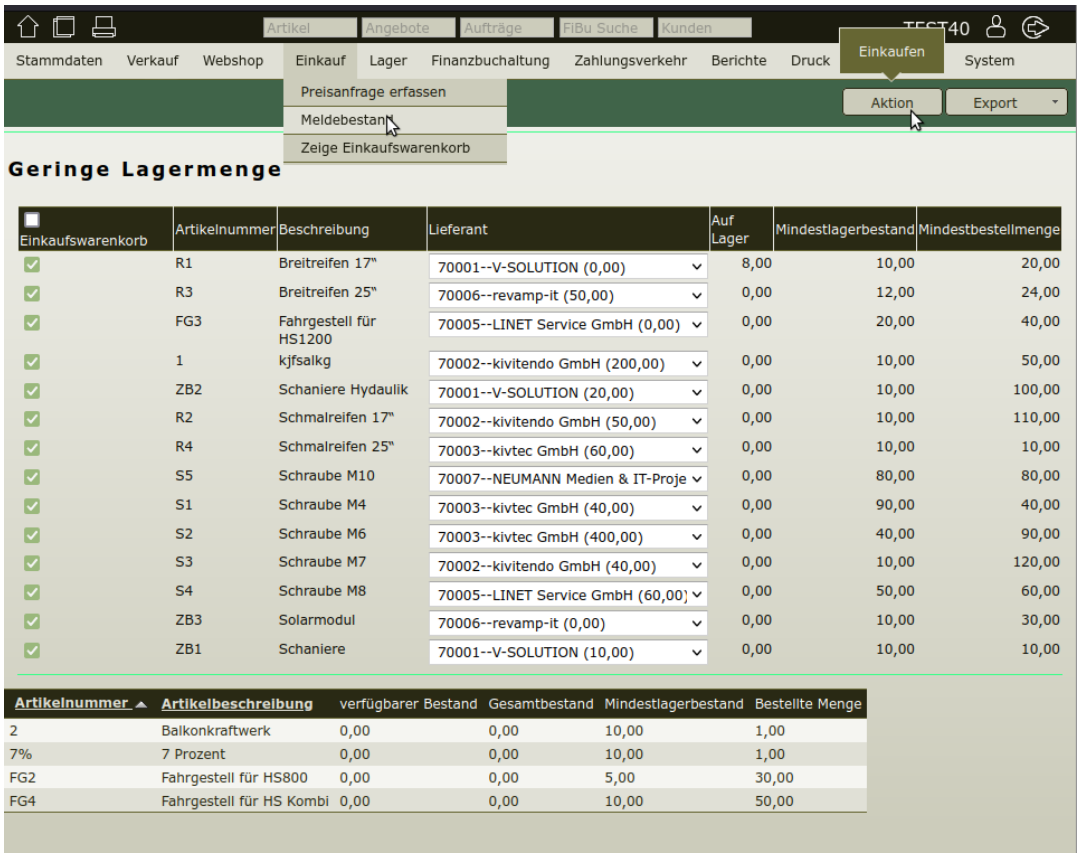
3.12.1. So kommen die Artikel in den Einkaufswaren-korb:

1) Sofern ein Artikel einen oder mehrere Lieferanten hat kann dieser über den Button „Einkaufen“ in den Einkaufswarenkorn gelegt werden. Soll der Einkauf über den Mindestlagerbestand gesteuert werden sind die Felder "Mindestlagerbestand" und "Bestellmenge" auszufüllen.



Der Button „Einkaufen“ ist deaktiviert, wenn es keinen Lieferanten zu diesem Artikel gibt oder der Artikel schon im Einkaufs-warenkorb liegt.

2) Bericht „Meldebestand“ Über das Menu "Einkauf->Meldebestand" kann der Meldebestand abgerufen werden

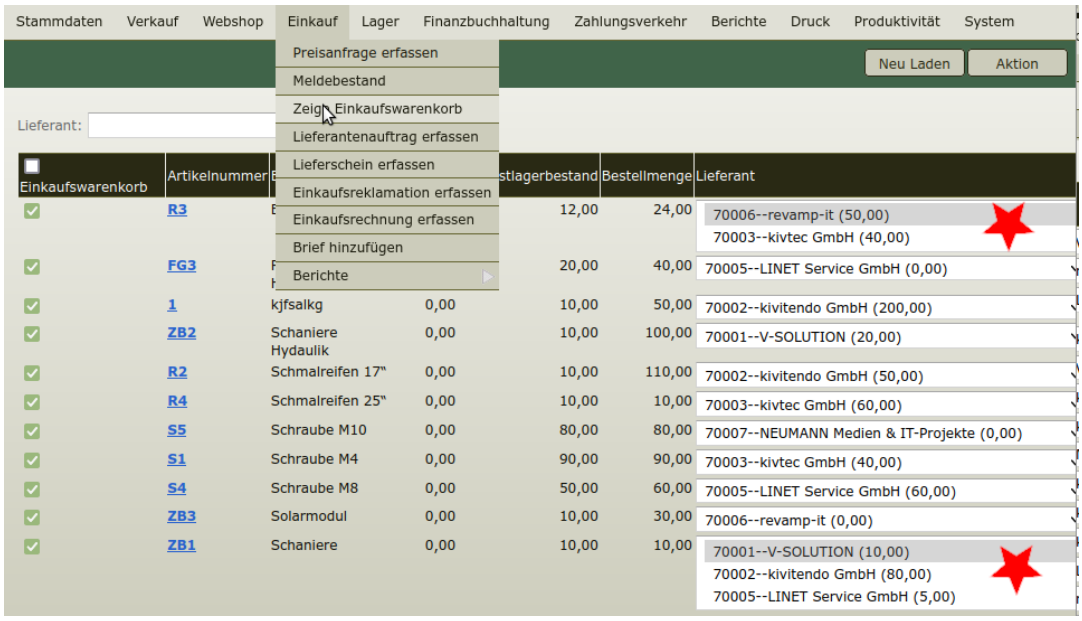


Die obere Tabelle zeigt die Artikel an, die den Mindestbestand unterschritten haben und können sofern markiert über die Aktion "Einkaufen" in den Einkaufswarenkorb gelegt werden. Hierzu muss das Feld Mindesbestand in den Artikelstammdaten einen Wert > 0 haben

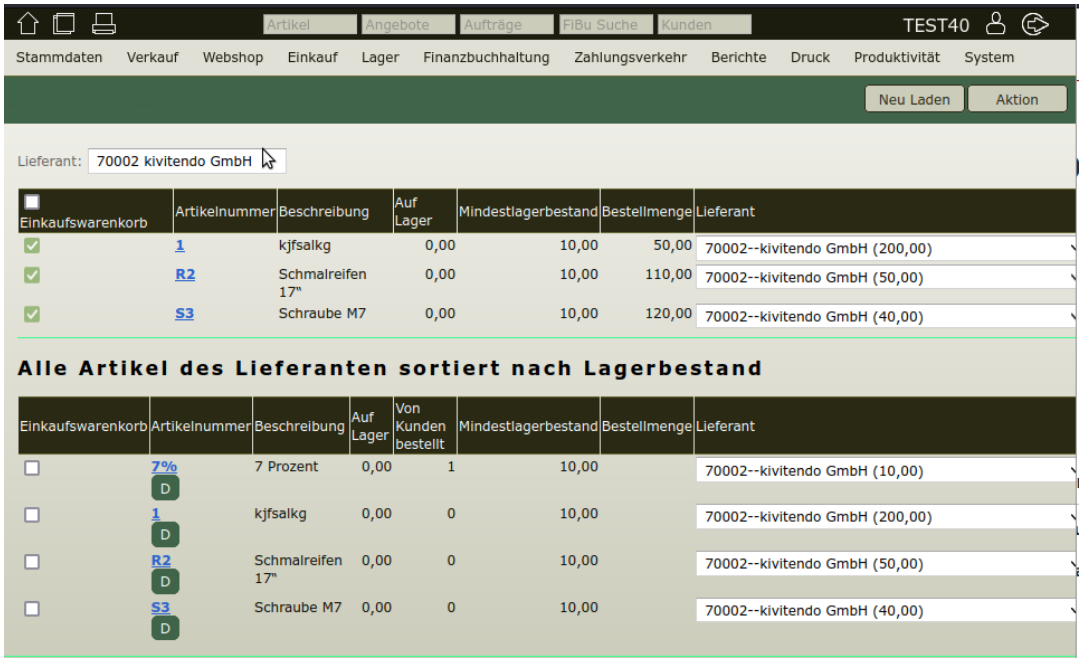
Die untere Tabelle zeigt die Artikel an, die den Mindestbestand unterschritten haben aber bestellt sind.

3.12.2. Der Einkaufswarenkorb

Ohne Filter: der erste Lieferant ist ausgewählt. Hier kann jetzt auch ein anderer Lieferant ausgewählt werden.



Wird dann nach einem Lieferanten gefiltert und dieser ist und bleibt eindeutig, kann daraus ein Lieferantenauftrag erstellt werden.



Die obere Tabelle zeigt die Artikel an, die den Mindestbestand unterschritten haben, der untere Teil zeigt alle Artikel des Lieferanten an, somit ist es möglich diese Artikel auszuwählen und gleich mitzubestellen

3.13. Inventur-Zählung anlegen und abgleichen

Im Gegensatz zum Anlegen von Inventurbuchungen über Lager->Inventur eigenen sich Inventur-Zählungen für die Datenerfassung per Code-Scanner, da zuerst alle Zählergebnisse gesammelt und dann alle gemeinsam in einem Schritt per Lagertransaktion abgeglichen werden.

Das Erfassen einer Inventurzählung erfolgt im Menü System → Inventur-Zählungen. Inventur-Zählungen lassen sich auch auf einzelne Lagerplätze einschränken.

Das eigentliche Zählen erfolgt per Desktop- oder Mobilgerät über den Menüpunkt Lager → Inventur-Zählung. In der Deskop-Variante werden Lagerplatz, wahlweise Artikel oder EAN, Menge und optional die Chargennummer eingegeben. In der Mobil-Variante ist aktuell nur die Eingabe einer EAN als Barcode per Scanner möglich.

Sobald alle Lagerposten für die Inventur-Zählung gezählt wurden, lässt sich diese abgeglichen. Dabei wird zunächst ein Bericht über Ist- und Soll-Werte der Zählung zur Kontrolle angezeigt. Ebenfalls im Bericht enthalten ist der Lagerabgang oder -zufluss seit dem Erfassen der Inventur-Zählung. Bei erfolgter Bestätigung erfolgen die notwendigen Lager-Korrekturbuchungen, um die Soll-Werte in kivitendos Lagerbestand den Ist-Werten der physisch gezählten Posten anzugleichen.

3.14. Zeiterfassung

Seit der Version 3.5.7 enthält kivitendo ein Modul zur Zeiterfassung. Damit ist es möglich, auftrags-, kunden- oder projektbezogen, Arbeitszeiten zu erfassen. Die erfassten Zeiten können über einen Hintergrund-Job in Lieferscheine umgewandelt werden.

3.14.1. Konfiguration

Die Zeiterfasssung funktioniert auch ohne Konfiguration, sofern der Benutzer die entsprechenden Rechte besitzt. Allerdings ist es möglich, Artikel für die Zeiterfassung zu konfigurieren, die im Bericht ausgewertet und bei der Umwandlung zum Lieferschein verwendet werden können. Zudem kann auf benutzerebene eingestellt werden, ob Zeiten mit Start- und End-Zeit oder mit Datum und Dauer erfasst werden sollen.

3.14.1.1. Zugriffsrechte

In der Rechteverwaltung im Admin-Bereich können im Zusammenhang mit der Zeiterfassung drei verschiedene Rechte vergeben werden:

- *Zeiterfassungen erfassen, bearbeiten und ansehen*

  Diese Recht steuert, ob eine Gruppe den Menüpunkt zur Zeiterfassung überhaupt sehen und diese verwenden kann.

- *Zeiterfassungseinträge aller Mitarbeiter anzeigen*

  Ohne dieses Recht darf ein Benutzer einer Benutzergruppe, die die Zeiterfassung verwenden darf, nur Zeiteinträge im Bericht sehen, bei denen dieser als Mitarbeiter eingetragen ist.

- *Zeiterfassungseinträge aller Mitarbeiter bearbeiten*

  Ohne dieses Recht darf ein Benutzer einer Benutzergruppe, die die Zeiterfassung verwenden darf, nur Zeiteinträge anlegen und bearbeiten, bei denen dieser als Mitarbeiter eingetragen ist.

3.14.1.2. Artikel für Zeiterfassung

Unter *System → Artikel für Zeiterfassung* können Artikel zur Nutzung mit der Zeiterfassung erfasst werden. Diese Artikel müssen eine zeitbasierte Einheit haben.

3.14.1.3. Benutzereinstellungen

Unter *Programm → Benutzereinstellungen* im Reiter *Persönliche Einstellungen* bei *Datum und Dauer für Zeiterfassung verwenden* kann der Benutzer angeben, ob Zeiten mit Start- und End-Zeit oder mit Datum und Dauer erfasst werden sollen.

3.14.2. Erfassen

Über den Menüpunkt *Produktivität → Zeiterfassung* gelangt man zur Erfassungsmaske der Zeiterfassung. Je nach Einstellung muss ein Datum oder Datum und Start-Zeit angegeben werden. Ebenso sind die Felder *Kunde* und *Beschreibung* Pflichtfelder.

Wird ein Auftrag oder ein Projekt, dem ein Kunde zugeordnet ist, ausgewählt, so wird das Feld *Kunde* automatisch mit dem zugehörigen Kunden gefüllt und für die manuelle Bearbeitung gesperrt.

Die End-Zeit bzw. die Dauer ist kein Pflichtfeld, damit man einen Eintrag z.B. beim Start der Arbeit anlegen und später am Ende dann die End-Zeit bzw. Dauer erfassen kann. Bei Eingabe über Start- und End-Zeit dürfen sich die Zeiten für ein und denselben Mitarbeiter nicht überlappen.

Falls der Bearbeiter das Recht *Zeiterfassungseinträge aller Mitarbeiter bearbeiten* besitzt, kann auch der Mitarbeiter für diesen Zeiteintrag ausgewählt werden.

3.14.3. Bericht

Ein Bericht über die erfassten Zeiten lässt sich über *Produktivität → Berichte → Zeiterfassung* ausgeben. Hier lassen sich die Zeiten auch filtern, sortieren und exportieren.

Ausgewählte Zeiteinträge können über *Aktionen → Als gebucht markieren* als schon gebucht markiert werden. Dies geschieht normalerweise bei der Konvertierung zum Lieferschein und kann hier händisch durchgeführt werden, wenn diese Zeitbuchungen bei der Konvertierung nicht mehr berücksichtigt werden sollen.

3.14.4. Konvertierung zu Lieferscheinen

Über den Hintergrund-Job *ConvertTimeRecordings* können Zeiteinträge in Lieferscheine umgewandelt werden. Hierbei werden alle noch nicht gebuchten Zeiteinträge eines bestimmten Zeitraums für bestimmte Kunden gesammelt und in Lieferscheine

umgewandelt. Pro Kunde wird ein Lieferschein generiert und pro Artikel wird eine Positionszeile erzeugt. Dabei werden Einträge mit gleichen Beschreibungen zusammengefasst. Die Details landen im Langtext der Position.

Ohne übergebene Parameter im Feld *Daten* werden alle Zeiteinträge des letzten Monats für alle Kunden umgewandelt. Dabei werden die Zeiten auf volle Viertelstunden aufgerundet (siehe auch Konfigurations-Optionen der Zeiterfassung [96]).

3.14.4.1. Konfigurations-Optionen

Zur generellen Konfiguration von Hintergrund-Jobs und zur Übergabe von Parametern als Daten an diese, siehe Abschnitt 2.8, „Der Task-Server“ [17] bzw. Abschnitt 2.9, „Konfiguration der Hintergrund-Jobs“ [19]).

Folgende Parameter können als *Daten* an den Hintergrund-Job zur Konvertierung übergeben werden:

- *from_date*

  Das Startdatum, von welchen an die Zeiteinträge berücksicht werden. Voreinstellung ist der erste Tag des vorherigen Montats.

  Beispiel (das Zeitformat hängt von der persönlichen Einstellungen ab):

  ```
  from_date: 01.12.2020
  ```

- *to_date*

  Das Datum, bis zu welchem die Zeiteinträge zur Konvertierung gesammelt werden. Voreinstellung ist der letzte Tag des vorherigen Montas.

  Beispiel (das Zeitformat hängt von der persönlichen Einstellungen ab):

  ```
  to_date: 15.12.2020
  ```

- *customernumbers*

  Eine Liste mit Kundennummern, für die Zeiteinträge gesammelt werden sollen. Wird diese Liste nicht angegeben, werden Einträge für alle Kunden berücksichtig.

  ```
  customernumbers: [c1,22332,334343]
  ```

- *override_part_id*

  Die Datenbank-Id einer zeitbasierten Dienstleistung, die verwendet werden soll, um die Zeiteinträge zu buchen. Dieser Wert überschreibt den Eintrag des Artikels des Zeiteintrags.

- *default_part_id*

  Die Datenbank-Id einer zeitbasierten Dienstleistung, die verwendet werden soll, um die Zeiteinträge zu buchen, falls im Zeiteintrag kein Artikel gespeichert ist.

- *rounding*

  Ist dieser Wert `0`, so werden die Zeiten nicht gerundet. Ist der Wert `1` (oder "wahr" in Perl"), so werden die Zeiten auf volle Viertelstunden aufgerundet, also z.B. 0.25h, 0.5h, 0.75h, 1.25h ...

  Voreinstellung ist `1` ("wahr").

- *link_order*

  Ist der Wert `1` (oder "wahr" in Perl"), so verknüpft der Hintergrund-Job den erzeugten Lieferschein mit dem im Zeiteintrag angegebenen Auftrag. Ist kein Auftrag angegeben, wird versucht, einen Auftrag für den angegebenen Kunden und die angegebene Projektnummer zu finden.

  Folgende Kriterien werden zum Finden eines Vorgängerauftrags angwendet:

- Auftrag ist im Zeiteintrag angegeben *oder*
- Datenbank-Id des Projekts des Auftrags ist gleich der Datenbank-Id des Zeiteintrags *oder* der in den Daten übergebenen Projekt-Id
- Kunde des Auftrags ist gleich Kunde des Zeiteintrags
- der Auftrag muss mindestens eine zugehörige zeitbasierte Position haben
- das Projekt muss gültig und aktiv sein

Voreinstellung ist aus. Wenn dieser Parameter gesetzt ist, läuft der Hintergrund-Job auf einen Fehler, falls kein geeigneter Vorgängerauftrag gefunden werden kann.

Der Job beachtet nicht, ob der Auftrag schon geliefert oder geschlossen ist. Wenn der Kundenauftrag überliefert wird, muss dies organisatorisch geklärt werden. Der Kundenauftrag kann auch bereits geschlossen sein, d.h. der Betrag ist vollständig fakturiert, aber die Leistungen sind noch nicht vollständig geliefert (einfacher Fall: "Vorauskasse").

siehe auch den Hintergrund-Job `CloseProjectsBelongingToClosedSalesOrder` für eine weitergehende Automatisierung der Abläufe.

- *override_project_id*

  Verwende diese Datenbank-Id für das Projekt an Stelle des im Zeiteintrag angegebenen Projekts, um den zugehörigen Auftrag zu finden. Dieser Parameter wird nur berücksichtig, wenn *link_order* wahr ist.

- *default_project_id*

  Verwende diese Datenbank-Id für das Projekt, falls im Zeiteintrag kein Projekt angegeben ist, um den zugehörigen Auftrag zu finden. Dieser Parameter wird nur berücksichtig, wenn

3.15. Ticketsystem-Schnittstelle

kivitendo kann in den Kunden- und Lieferanten-Stammdaten zugehörige Tickets aus einem Ticketsystem (Issue Tracker) abrufen und diese in einem Tab anzeigen. Dazu ist in der Mandantenkonfiguration -> Features -> Experimentelle Features -> Ticket System Anbieter zunächst ein Anbieter auszuwählen. Ferner ist anbieterabhängig weitere Konfiguration erforderlich.

3.15.1. Atlassian Jira

Diese Implementierung funktioniert mit Atlassian Jira Cloud. (Die on-premises Variante von Jira hat eine andere API.)

Zu den Tickets werden folgende sortierbare Spalten angezeigt: Schlüssel, Zusammenfassung, Priorität, Status, Ersteller, Zugewiesen, Erstellt am, Erneuert am. Über den Link in der Spalte Schlüssel kann zu Jira gesprungen werden.

Implementiert ist Authentifizierung via OAuth2 Tokens. In Jira muss kivitendo zunächst als App angelegt werden. Dabei werden `client_id`, `client_secret` und `redirect_uri` vergeben. Zu beachten ist, dass `redirect_uri` diejenige URL sein muss, unter welcher kivitendo zum Anlegen der OAuth-Tokens vom Benutzer aufgerufen wird, gefolgt vom Endpunkt `/oauth.pl`. Alle drei Parameter sind in der `kivitendo.conf` im Abschnitt `[oauth2_atlassian_jira]` einzustellen, wie unten exemplarisch gezeigt ist.

Die App muss in Jira mit einem Benutzer oder mit mehreren Benutzern verknüpft werden:

- ein einziger Jira-Benutzer, dessen Token mandantenweit alle kivitendo Benutzer zum Abruf von Tickets nutzen; oder
- jeder kivitendo Benutzer bezieht sein eigenes Token für Jira, welches nur er selbst nutzen darf, um Tickets abzurufen

Die OAuth Tokens werden in kivitendo unter Programm -> OAuth Tokens angelegt. Ist in kivitendo ein mandantenweites Token und ein nutzerbezogenes Token für einen bestimmten Nutzer vorhanden, so wird das nutzerbezogene Token bevorzugt.

Nicht notwendig, aber für die Geschwindigkeit der Abfragen vorteilhaft, ist es, die `cloud_id` und `cloud_url` der gewünschten Jira Instanz ebenfalls in der `kivitendo.conf` einzustellen. (Zum Finden dieser Parameter nutzen Entwickler die Konsole und rufen `accessible_resources` aus `SL/TicketSystem/Jira.pm` auf.) Fehlt diese Angabe, so wird die von Jira als erstes zurückgegebene Cloud verwendet, wozu für jede Abfrage ein zweiter HTTP-Request nötig ist.

Exemplarischer Konfigurationsabschnitt in `kivitendo.conf`:

```
[oauth2_atlassian_jira]

client_id     = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
client_secret = xxxxxxx-xxxxx-xxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxxxxxxxx-xxxxxxxxxxxxxxxxxxxx
redirect_uri  = http://kiviserver.lan/oauth.pl

# optional:
[atlassian_jira]

cloud_id  = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
cloud_url = https://firma.atlassian.net
```

Beim Kopieren einer Produktivdatenbank für Testzwecke sollten die OAuth Tokens ausgelassen werden, da beim Refresh im Testsystem die Tokens im Produktivsystem ungültig würden. Siehe hierzu die Option `--exclude-table-data=oauth_token` in `man pg_dump(1)`.

4 Entwicklerdokumentation

4.1. Globale Variablen

4.1.1. Wie sehen globale Variablen in Perl aus?

Globale Variablen liegen in einem speziellen namespace namens "main", der von überall erreichbar ist. Darüber hinaus sind bareword globs global und die meisten speziellen Variablen sind... speziell.

Daraus ergeben sich folgende Formen:

`$main::form`
: expliziter Namespace "main"

`$::form`
: impliziter Namespace "main"

`open FILE, "file.txt"`
: `FILE` ist global

`$_`
: speziell

Im Gegensatz zu PHP™ gibt es kein Schlüsselwort wie "`global`", mit dem man importieren kann. `my`, `our` und `local` machen was anderes.

`my $form`
: lexikalische Variable, gültig bis zum Ende des Scopes

`our $form`
: `$form` referenziert ab hier `$PACKAGE::form`.

`local $form`
: Alle Änderungen an `$form` werden am Ende des scopes zurückgesetzt

4.1.2. Warum sind globale Variablen ein Problem?

Das erste Problem ist FCGI™.

SQL-Ledger™ hat fast alles im globalen namespace abgelegt, und erwartet, dass es da auch wiederzufinden ist. Unter FCGI™ müssen diese Sachen aber wieder aufgeräumt werden, damit sie nicht in den nächsten Request kommen. Einige Sachen wiederum sollen nicht gelöscht werden, wie zum Beispiel Datenbankverbindungen, weil die sehr lange zum Initialisieren brauchen.

Das zweite Problem ist `strict`. Unter `strict` werden alle Variablen die nicht explizit mit `Package`, `my` oder `our` angegeben werden als Tippfehler angemarkert, dies hat, seit der Einführung, u.a. schon so manche langwierige Bug-Suche verkürzt. Da globale Variablen aber implizit mit Package angegeben werden, werden die nicht geprüft, und somit kann sich schnell ein Tippfehler einschleichen.

4.1.3. Kanonische globale Variablen

Um dieses Problem im Griff zu halten gibt es einige wenige globale Variablen, die kanonisch sind, d.h. sie haben bestimmte vorgegebenen Eigenschaften, und alles andere sollte anderweitig umhergereicht werden.

Diese Variablen sind im Moment die folgenden neun:

- `$::form`
- `%::myconfig`
- `$::locale`
- `$::lxdebug`
- `$::auth`
- `$::lx_office_conf`
- `$::instance_conf`
- `$::dispatcher`
- `$::request`

Damit diese nicht erneut als Müllhalde missbraucht werden, im Folgenden eine kurze Erläuterung der bestimmten vorgegebenen Eigenschaften (Konventionen):

4.1.3.1. $::form

- Ist ein Objekt der Klasse "`Form`"
- Wird nach jedem Request gelöscht
- Muss auch in Tests und Konsolenscripts vorhanden sein.
- Enthält am Anfang eines Requests die Requestparameter vom User
- Kann zwar intern über Requestgrenzen ein Datenbankhandle cachen, das wird aber momentan absichtlich zerstört

`$::form` wurde unter SQL Ledger™ als Gottobjekt für alles missbraucht. Sämtliche alten Funktionen unter SL/ mutieren `$::form`, das heißt, alles was einem lieb ist (alle Variablen die einem ans Herz gewachsen sind), sollte man vor einem Aufruf (!) von zum Beispiel `IS->retrieve_customer()` in Sicherheit bringen.

Z.B. das vom Benutzer eingestellte Zahlenformat, bevor man Berechnung in einem bestimmten Format durchführt (SL/Form.pm Zeile 3552, Stand version 2.7beta), um dies hinterher wieder auf den richtigen Wert zu setzen:

```
my $saved_numberformat    = $::myconfig{numberformat};
$::myconfig{numberformat} = $numberformat;
# (...) div Berechnungen
$::myconfig{numberformat} = $saved_numberformat;
```

Das Objekt der Klasse Form hat leider im Moment noch viele zentrale Funktionen die vom internen Zustand abhängen, deshalb bitte nie einfach zerstören oder überschreiben (zumindestens nicht kurz vor einem Release oder in Absprache über bspw. die devel-Liste ;-). Es geht ziemlich sicher etwas kaputt.

`$::form` ist gleichzeitig der Standard Scope in den Template::Toolkit™ Templates außerhalb der Controller: der Ausdruck `[% var %]` greift auf `$::form->{var}` zu. Unter Controllern ist der Standard Scope anders, da lautet der Zugriff `[% FORM.var %]`. In Druckvorlagen sind normale Variablen ebenfall im `$::form` Scope, d.h. `<%var%>` zeigt auf `$::form->{var}`. Nochmal von der anderen Seite erläutert, innerhalb von (Web-)Templates sieht man häufiger solche Konstrukte:

```
[%- IF business %]
# (... Zeig die Auswahlliste Kunden-/Lieferantentyp an)
```

```
[%- END %]
```

Entweder wird hier dann $::form->{business} ausgewertet oder aber der Funktion `$form->parse_html_template` wird explizit noch ein zusätzlicher Hash übergeben, der dann auch in den (Web-)Templates zu Verfügung steht, bspw. so:

```
$form->parse_html_template("is/form_header", \%TMPL_VAR);
```

Innerhalb von Schleifen wird `$::form->{TEMPLATE_ARRAYS}{var}[$index]` bevorzugt, wenn vorhanden. Ein Beispiel findet sich in SL/DO.pm, welches über alle Positionen eines Lieferscheins in Schleife läuft:

```
for $i (1 .. $form->{rowcount}) {
  # ...
  push @{ $form->{TEMPLATE_ARRAYS}{runningnumber} },   $position;
  push @{ $form->{TEMPLATE_ARRAYS}{number} },          $form->{"partnumber_$i"};
  push @{ $form->{TEMPLATE_ARRAYS}{description} },     $form->{"description_$i"};
  # ...
}
```

4.1.3.2. %::myconfig

- Das einzige Hash unter den globalen Variablen
- Wird spätestens benötigt wenn auf die Datenbank zugegriffen wird
- Wird bei jedem Request neu erstellt.
- Enthält die Userdaten des aktuellen Logins
- Sollte nicht ohne Filterung irgendwo gedumpt werden oder extern serialisiert werden, weil da auch der Datenbankzugriff für diesen user drinsteht.
- Enthält unter anderem Datumsformat dateformat und Nummernformat numberformat
- Enthält Datenbankzugriffinformationen

`%::myconfig` ist im Moment der Ersatz für ein Userobjekt. Die meisten Funktionen, die etwas anhand des aktuellen Users entscheiden müssen, befragen `%::myconfig`. Innerhalb der Anwendungen sind dies überwiegend die Daten, die sich unter Programm -> Einstellungen befinden, bzw. die Informationen über den Benutzer die über die Administrator-Schnittstelle eingegeben wurden.

4.1.3.3. $::locale

- Objekt der Klasse "Locale"
- Wird pro Request erstellt
- Muss auch für Tests und Scripte immer verfügbar sein.
- Cached intern über Requestgrenzen hinweg benutzte Locales

Lokalisierung für den aktuellen User. Alle Übersetzungen, Zahlen- und Datumsformatierungen laufen über dieses Objekt.

4.1.3.4. $::lxdebug

- Objekt der Klasse "LXDebug"
- Wird global gecached
- Muss immer verfügbar sein, in nahezu allen Funktionen

`$::lxdebug` stellt Debuggingfunktionen bereit, wie "`enter_sub`" und "`leave_sub`", mit denen in den alten Modulen ein brauchbares Tracing gebaut ist, "`log_time`", mit der man die Wallclockzeit seit Requeststart loggen kann, sowie "`mes-`

`sage`" und "`dump`" mit denen man flott Informationen ins Log (users/kivitendo-debug.log relativ zum kivitendo-Installationsverzeichnis) packen kann.

Beispielsweise so:

```
$main::lxdebug->message(0, 'Meine Konfig:' . Dumper (%::myconfig));
$main::lxdebug->message(0, 'Wer bin ich? Kunde oder Lieferant:' . $form->{vc});
```

4.1.3.5. $::auth

- Objekt der Klasse "SL::Auth"
- Wird global gecached
- Hat eine permanente DB Verbindung zur Authdatenbank
- Wird nach jedem Request resettet.

`$::auth` stellt Funktionen bereit um die Rechte des aktuellen Users abzufragen. Obwohl diese Informationen vom aktuellen User abhängen wird das Objekt aus Geschwindigkeitsgründen nur einmal angelegt und dann nach jedem Request kurz resettet.

Dieses Objekt kapselt auch den gerade aktiven Mandanten. Dessen Einstellungen können über `$::auth->client` abgefragt werden; Rückgabewert ist ein Hash mit den Werten aus der Tabelle `auth.clients`.

4.1.3.6. $::lx_office_conf

- Objekt der Klasse "`SL::LxOfficeConf`"
- Global gecached
- Repräsentation der `config/kivitendo.conf[.default]`-Dateien

Globale Konfiguration. Configdateien werden zum Start gelesen und danach nicht mehr angefasst. Es ist derzeit nicht geplant, dass das Programm die Konfiguration ändern kann oder sollte.

Beispielsweise ist über den Konfigurationseintrag [debug] die Debug- und Trace-Log-Datei wie folgt konfiguriert und verfügbar:

```
[debug]
file_name = users/kivitendo-debug.log
```

ist der Key `file` im Programm als `$::lx_office_conf->{debug}{file}` erreichbar.

Warnung

Zugriff auf die Konfiguration erfolgt im Moment über Hashkeys, sind also nicht gegen Tippfehler abgesichert.

4.1.3.7. $::instance_conf

- Objekt der Klasse "`SL::InstanceConfiguration`"
- wird pro Request neu erstellt

Funktioniert wie `$::lx_office_conf`, speichert aber Daten die von der Instanz abhängig sind. Eine Instanz ist hier eine Mandantendatenbank. Beispielsweise überprüft

```
$::instance_conf->get_inventory_system eq 'perpetual'
```

ob die berüchtigte Bestandsmethode zur Anwendung kommt.

4.1.3.8. $::dispatcher

- Objekt der Klasse "`SL::Dispatcher`"

- wird pro Serverprozess erstellt.
- enthält Informationen über die technische Verbindung zum Server

Der dritte Punkt ist auch der einzige Grund warum das Objekt global gespeichert wird. Wird vermutlich irgendwann in einem anderen Objekt untergebracht.

4.1.3.9. $::request

- Hashref (evtl später Objekt)
- Wird pro Request neu initialisiert.
- Keine Unterstruktur garantiert.

`$::request` ist ein generischer Platz um Daten "für den aktuellen Request" abzulegen. Sollte nicht für action at a distance benutzt werden, sondern um lokales memoizing zu ermöglichen, das garantiert am Ende des Requests zerstört wird.

Vieles von dem, was im moment in `$::form` liegt, sollte eigentlich hier liegen. Die groben Differentialkriterien sind:

- Kommt es vom User, und soll unverändert wieder an den User? Dann `$::form`, steht da eh schon
- Sind es Daten aus der Datenbank, die nur bis zum Ende des Requests gebraucht werden? Dann `$::request`
- Muss ich von anderen Teilen des Programms lesend drauf zugreifen? Dann `$::request`, aber Zugriff über Wrappermethode

4.1.4. Ehemalige globale Variablen

Die folgenden Variablen waren einmal im Programm, und wurden entfernt.

4.1.4.1. $::cgi

- war nötig, weil cookie Methoden nicht als Klassenfunktionen funktionieren
- Aufruf als Klasse erzeugt Dummyobjekt was im Klassennamespace gehalten wird und über Requestgrenzen leaked
- liegt jetzt unter `$::request->{cgi}`

4.1.4.2. $::all_units

- war nötig, weil einige Funktionen in Schleifen zum Teil ein paar hundert mal pro Request eine Liste der Einheiten brauchen, und de als Parameter durch einen Riesenstack von Funktionen geschleift werden müssten.
- Liegt jetzt unter `$::request->{cache}{all_units}`
- Wird nur in `AM->retrieve_all_units()` gesetzt oder gelesen.

4.1.4.3. %::called_subs

- wurde benutzt um callsub deep recursions abzufangen.
- Wurde entfernt, weil callsub nur einen Bruchteil der möglichen Rekursioenen darstellt, und da nie welche auftreten.
- komplette recursion protection wurde entfernt.

4.2. Entwicklung unter FastCGI

4.2.1. Allgemeines

Wenn Änderungen in der Konfiguration von kivitendo gemacht werden, muss der Webserver neu gestartet werden.

Bei der Entwicklung für FastCGI ist auf ein paar Fallstricke zu achten. Dadurch, dass das Programm in einer Endlosschleife läuft, müssen folgende Aspekte beachtet werden.

4.2.2. Programmende und Ausnahmen

Betrifft die Funktionen `warn`, `die`, `exit`, `carp` und `confess`.

Fehler, die dass Programm normalerweise sofort beenden (fatale Fehler), werden mit dem FastCGI Dispatcher abgefangen, um das Programm am Laufen zu halten. Man kann mit `die`, `confess` oder `carp` Fehler ausgeben, die dann vom Dispatcher angezeigt werden. Die kivitendo eigene `$::form-`error()> tut im Prinzip das Gleiche, mit ein paar Extraoptionen. `warn` und `exit` hingegen werden nicht abgefangen. `warn` wird direkt nach STDERR, also in Server Log eine Nachricht schreiben (sofern in der Konfiguration nicht die Warnungen in das kivitendo Log umgeleitet wurden), und `exit` wird die Ausführung beenden.

Prinzipiell ist es kein Beinbruch, wenn sich der Prozess beendet, fcgi wird ihn sofort neu starten. Allerdings sollte das die Ausnahme sein. Quintessenz: Bitte kein `exit` benutzen, alle anderen Exceptionmechanismen sind ok.

4.2.3. Globale Variablen

Um zu vermeiden, dass Informationen von einem Request in einen anderen gelangen, müssen alle globalen Variablen vor einem Request sauber initialisiert werden. Das ist besonders wichtig im `$::cgi` und `$::auth` Objekt, weil diese nicht gelöscht werden pro Instanz, sondern persistent gehalten werden.

In `SL::Dispatcher` gibt es einen sauber abgetrennten Block, der alle kanonischen globalen Variablen listet und erklärt. Bitte keine anderen einführen ohne das sauber zu dokumentieren.

Datenbankverbindungen wird noch ein Guide verfasst werden, wie man sicher geht, dass man die richtige erwischt.

4.2.4. Performance und Statistiken

Die kritischen Pfade des Programms sind die Belegmasken, und unter diesen ganz besonders die Verkaufsrechnungsmaske. Ein Aufruf der Rechnungsmaske in kivitendo 2.4.3 stable dauert auf einem Core2duo mit 4GB Arbeitsspeicher und Ubuntu 9.10 eine halbe Sekunde. In der 2.6.0 sind es je nach Menge der definierten Variablen 1-2s. Ab der Moose/Rose::DB Version sind es 5-6s.

Mit FastCGI ist die neuste Version auf 0,26 Sekunden selbst in den kritischen Pfaden, unter 0,15 sonst.

4.3. Entwicklung unter CGI

Der Einsatz von CGI (in Abgrezung zu FastCGI) ist in aller Regel performancetechnisch nachteilig. Er kann allerdings für Debugging-Zwecke sinnvoll sein. Folgender Konfigurationsschnippsel funktioniert mit Apache:

```
AddHandler  cgi-script .pl
AliasMatch ^/url/for/kivitendo-erp/[^/]+\.pl$ /var/www/kivitendo-erp/cgi/dispatcher.pl
# Alias     /url/for/kivitendo-erp/webdav     /var/www/kivitendo-erp/webdav
Alias       /url/for/kivitendo-erp            /var/www/kivitendo-erp/public

<Directory "/var/www/kivitendo-erp/cgi">
  Options +ExecCGI
  Require all granted
</Directory>

<Directory "/var/www/kivitendo-erp/public">
  Options -ExecCGI +FollowSymlinks
  Require all granted
</Directory>
```

4.4. Programmatische API-Aufrufe

4.4.1. Einführung

Es ist möglich, Funktionen in kivitendo programmatisch aus anderen Programmen aufzurufen. Dazu ist nötig, dass Authentifizierungsinformationen in jedem Aufruf mitgegeben werden. Dafür gibt es die HTTP-»Basic«-Authentifizierung. Der zu verwendende Mandant ist der Standard-Mandant.

4.4.2. HTTP-»Basic«-Authentifizierung

Für diese Methode muss jedem Request der bekannte HTTP-Header `Authorization` mitgeschickt werden (siehe RFC 7617[1]). Unterstützt wird ausschließlich die »Basic«-Methode. Loginname und Passwort werden bei dieser Methode durch einen Doppelpunkt getrennt und Base64-encodiert im genannten HTTP-Header übertragen.

Diese Informationen müssen einen vorhandenen Account benennen. kivitendo prüft genau wie bei Benutzung über den Webbrowser, ob dieser Account Zugriff auf den Mandanten sowie auf die angeforderte Funktion hat.

Da die Logininformationen im Klartext im Request stehen, sollte der Zugriff auf kivitendo ausschließlich über HTTPS verschlüsselt erfolgen.

4.4.3. Beispiele

Das folgende Beispiel nutzt das Kommandozeilenprogramm »curl« und ruft die Funktion auf, die eine vorhandene Telefonnummer in den Ansprechpersonen sucht und dazu Informationen zurückliefert. Dabei wird die HTTP-»Basic«-Authentifizierung genutzt.

```
$ curl --silent --user 'jdoe:SecretPassword!' \
  'https://…/controller.pl?action=PhoneNumber/look_up&number=053147110815'
```

4.5. SQL-Upgradedateien

4.5.1. Einführung

Datenbankupgrades werden über einzelne Upgrade-Scripte gesteuert, die sich im Verzeichnis `sql/Pg-upgrade2` befinden. In diesem Verzeichnis muss pro Datenbankupgrade eine Datei existieren, die neben den eigentlich auszuführenden SQL- oder Perl-Befehlen einige Kontrollinformationen enthält.

Kontrollinformationen definieren Abhängigkeiten und Prioritäten, sodass Datenbankscripte zwar in einer sicheren Reihenfolge ausgeführt werden (z.B. darf ein `ALTER TABLE` erst ausgeführt werden, wenn die Tabelle mit `CREATE TABLE` angelegt wurde), diese Reihenfolge aber so flexibel ist, dass man keine Versionsnummern braucht.

kivitendo merkt sich dabei, welches der Upgradescripte in `sql/Pg-upgrade2` bereits durchgeführt wurde und führt diese nicht erneut aus. Dazu dient die Tabelle "`schema_info`", die bei der Anmeldung automatisch angelegt wird.

4.5.2. Format der Kontrollinformationen

Die Kontrollinformationen sollten sich am Anfang der jeweiligen Upgradedatei befinden. Jede Zeile, die Kontrollinformationen enthält, hat dabei das folgende Format:

Für SQL-Upgradedateien:

```
-- @key: value
```

Für Perl-Upgradedateien:

[1] https://tools.ietf.org/html/rfc7617

```
# @key: value
```

Leerzeichen vor "`value`" werden entfernt.

Die folgenden Schlüsselworte werden verarbeitet:

`tag`
: Wird zwingend benötigt. Dies ist der "Name" des Upgrades. Dieser "tag" kann von anderen Kontrolldateien in ihren Abhängigkeiten verwendet werden (Schlüsselwort "`depends`"). Der "tag" ist auch der Name, der in der Datenbank eingetragen wird.

  Normalerweise sollte die Kontrolldatei genau so heißen wie der "tag", nur mit der Endung ".sql" bzw. "pl".

  Ein Tag darf nur aus alphanumerischen Zeichen sowie den Zeichen _ - ( ) bestehen. Insbesondere sind Leerzeichen nicht erlaubt und sollten stattdessen mit Unterstrichen ersetzt werden.

`charset`
: Empfohlen. Gibt den Zeichensatz an, in dem das Script geschrieben wurde, z.B. "`UTF-8`". Aus Kompatibilitätsgründen mit alten Upgrade-Scripten wird bei Abwesenheit des Tags für SQL-Upgradedateien der Zeichensatz "`ISO-8859-15`" angenommen. Perl-Upgradescripte hingegen müssen immer in UTF-8 encodiert sein und sollten demnach auch ein "`use utf8;`" enthalten.

`description`
: Benötigt. Eine Beschreibung, was in diesem Update passiert. Diese wird dem Benutzer beim eigentlichen Datenbankupdate angezeigt. Während der Tag in Englisch gehalten sein sollte, sollte die Beschreibung auf Deutsch erfolgen.

`depends`
: Optional. Eine mit Leerzeichen getrennte Liste von "tags", von denen dieses Upgradescript abhängt. kivitendo stellt sicher, dass die in dieser Liste aufgeführten Scripte bereits durchgeführt wurden, bevor dieses Script ausgeführt wird.

  Abhängigkeiten werden rekursiv betrachtet. Wenn also ein Script "b" existiert, das von Änderungen in "a" abhängt, und eine neue Kontrolldatei für "c" erstellt wird, die von Änderungen in "a" und "b" abhängt, so genügt es, in "c" nur den Tag "b" als Abhängigkeit zu definieren.

  Es ist nicht erlaubt, sich selbst referenzierende Abhängigkeiten zu definieren (z.B. "a" -> "b", "b" -> "c" und "c" -> "a").

`priority`
: Optional. Ein Zahlenwert, der die Reihenfolge bestimmt, in der Scripte ausgeführt werden, die die gleichen Abhängigkeitstiefen besitzen. Fehlt dieser Parameter, so wird der Wert 1000 benutzt.

  Dies ist reine Kosmetik. Für echte Reihenfolgen muss "depends" benutzt werden. kivitendo sortiert die auszuführenden Scripte zuerst nach der Abhängigkeitstiefe (wenn "z" von "y" abhängt und "y" von "x", so hat "z" eine Abhängigkeitstiefe von 2, "y" von 1 und "x" von 0. "x" würde hier zuerst ausgeführt, dann "y", dann "z"), dann nach der Priorität und bei gleicher Priorität alphabetisch nach dem "tag".

`ignore`
: Optional. Falls der Wert auf 1 (true) steht, wird das Skript bei der Anmeldung ignoriert und entsprechend nicht ausgeführt.

4.5.3. Format von in Perl geschriebenen Datenbankupgradescripten

In Perl geschriebene Datenbankscripte werden nicht einfach so ausgeführt sondern müssen sich an gewisse Konventionen halten. Dafür bekommen sie aber auch einige Komfortfunktionen bereitgestellt.

Ein Upgradescript stellt dabei eine vollständige Objektklasse dar, die vom Elternobjekt "`SL::DBUpgrade2::Base`" erben und eine Funktion namens "`run`" zur Verfügung stellen muss. Das Script wird ausgeführt, indem eine Instanz dieser Klasse erzeugt und darauf die erwähnte "`run`" aufgerufen wird.

Zu beachten ist, dass sich der Paketname der Datei aus dem Wert für "`@tag`" ableitet. Dabei werden alle Zeichen, die in Paketnamen ungültig wären (gerade Bindestriche), durch Unterstriche ersetzt. Insgesamt sieht der Paketname wie folgt aus: "`SL::DBUpgrade2::tag`".

Welche Komfortfunktionen zur Verfügung stehen, erfahren Sie in der Perl-Dokumentation zum oben genannten Modul; aufzurufen mit "**perldoc SL/DBUpgrade2/Base.pm**".

Ein Mindestgerüst eines gültigen Perl-Upgradescriptes sieht wie folgt aus:

```
# @tag: beispiel-upgrade-file42
# @description: Ein schönes Beispielscript
# @depends: release_3_1_0
package SL::DBUpgrade2::beispiel_upgrade_file42;

use strict;
use utf8;

use parent qw(SL::DBUpgrade2::Base);

sub run {
  my ($self) = @_;

  # hier Aktionen ausführen

  return 1;
}

1;
```

4.5.4. Hilfsscript dbupgrade2_tool.pl

Um die Arbeit mit den Abhängigkeiten etwas zu erleichtern, existiert ein Hilfsscript namens "`scripts/dbupgrade2_tool.pl`". Es muss aus dem kivitendo-ERP-Basisverzeichnis heraus aufgerufen werden. Dieses Tool liest alle Datenbankupgradescripte aus dem Verzeichnis `sql/Pg-upgrade2` aus. Es benutzt dafür die gleichen Methoden wie kivitendo selber, sodass alle Fehlersituationen von der Kommandozeile überprüft werden können.

Wird dem Script kein weiterer Parameter übergeben, so wird nur eine Überprüfung der Felder und Abhängigkeiten vorgenommen. Man kann sich aber auch Informationen auf verschiedene Art ausgeben lassen:

- Listenform: "**./scripts/dbupgrade2_tool.pl --list**"

  Gibt eine Liste aller Scripte aus. Die Liste ist in der Reihenfolge sortiert, in der kivitendo die Scripte ausführen würde. Es werden neben der Listenposition der Tag, die Abhängigkeitstiefe und die Priorität ausgegeben.

- Baumform: "**./scripts/dbupgrade2_tool.pl --tree**"

  Listet alle Tags in Baumform basierend auf den Abhängigkeiten auf. Die "Wurzelknoten" sind dabei die Scripte, von denen keine anderen abhängen. Die Unterknoten sind Scripte, die beim übergeordneten Script als Abhängigkeit eingetragen sind.

- Umgekehrte Baumform: "**./scripts/dbupgrade2_tool.pl --rtree**"

  Listet alle Tags in Baumform basierend auf den Abhängigkeiten auf. Die "Wurzelknoten" sind dabei die Scripte mit der geringsten Abhängigkeitstiefe. Die Unterknoten sind Scripte, die das übergeordnete Script als Abhängigkeit eingetragen haben.

- Baumform mit Postscriptausgabe: "**./scripts/dbupgrade2_tool.pl --graphviz**"

  Benötigt das Tool "**graphviz**", um mit seiner Hilfe die umgekehrte Baumform in eine Postscriptdatei namens "`db_dependencies.ps`" auszugeben. Dies ist vermutlich die übersichtlichste Form, weil hierbei jeder Knoten nur einmal ausgegeben wird. Bei den Textmodusbaumformen hingegen können Knoten und all ihre Abhängigkeiten mehrfach ausgegeben werden.

- Scripte, von denen kein anderes Script abhängt: "**./scripts/dbupgrade2_tool.pl --nodeps**"

  Listet die Tags aller Scripte auf, von denen keine anderen Scripte abhängen.

4.6. Translations and languages

4.6.1. Introduction

Anmerkung

Dieser Abschnitt ist in Englisch geschrieben, um internationalen Übersetzern die Arbeit zu erleichtern.

This section describes how localization packages in kivitendo are built. Currently the only language fully supported is German, and since most of the internal messages are held in English the English version is usable too.

4.6.2. Character set

All files included in a language pack must use UTF-8 as their encoding.

4.6.3. File structure

The structure of locales in kivitendo is:

```
kivitendo/locale/<langcode>/
```

where <langcode> stands for an abbreviation of the language package. The builtin packages use two letter ISO 639-1[2] codes, but the actual name is not relevant for the program and can easily be extended to IETF language tags[3] (i.e. "en_GB"). In fact the original language packages from SQL Ledger are named in this way.

In such a language directory the following files are recognized:

LANGUAGE
: This file is mandatory.

  The `LANGUAGE` file contains the self descripted name of the language. It should contain a native representation first, and in parenthesis an english translation after that. Example:

  ```
  Deutsch (German)
  ```

all
: This file is mandatory.

  The central translation file. It is essentially an inline Perl script autogenerated by **locales.pl**. To generate it, generate the directory and the two files mentioned above, and execute the following command:

  ```
  scripts/locales.pl <langcode>
  ```

  Otherwise you can simply copy one of the other languages. You will be told how many are missing like this:

  ```
  $ scripts/locales.pl en
  English - 0.6% - 2015/2028 missing
  ```

  A file named "`missing`" will be generated and can be edited. You can also edit the "`all`" file directly. Edit everything you like to fit the target language and execute **locales.pl** again. See how the missing words get fewer.

Num2text
: Legacy code from SQL Ledger. It provides a means for numbers to be converted into natural language, like `1523 => one thousand five hundred twenty three`. If you want to provide it, it must be inlinable Perl code which provides a `num2text` sub. If an `init` sub exists it will be executed first.

  Only used in the check and receipt printing module.

[2] http://en.wikipedia.org/wiki/ISO_639-1
[3] http://en.wikipedia.org/wiki/IETF_language_tag

special_chars

kivitendo comes with a lot of interfaces to different formats, some of which are rather picky with their accepted charset. The `special_chars` file contains a listing of chars not suited for different file format and provides substitutions. It is written in "Simple Ini" style, containing a block for every file format.

First entry should be the order of substitution for entries as a whitespace separated list. All entries are interpolated, so `\n`, `\x20` and `\\` all work.

After that every entry is a special char that should be translated when writing text into such a file.

Example:

```
[Template/XML]
order=& < > \n
&=&amp;
<=&lt;
>=&gt;
\n=<br>
```

Note the importance of the order in this example. Substituting < and > befor & would lead to $gt; become >

For a list of valid formats, see the German `special_chars` entry. As of this writing the following are recognized:

```
HTML
URL@HTML
Template/HTML
Template/XML
Template/LaTeX
Template/OpenDocument
filenames
```

The last of which is very machine dependent. Remember that a lot of characters are forbidden by some filesystems, for example MS Windows doesn't like ':' in its files where Linux doesn't mind that. If you want the files created with your language pack to be portable, find all chars that could cause trouble.

missing

This file is not a part of the language package itself.

This is a file generated by **scripts/locales.pl** while processing your locales. It's only to have the missing entries singled out and does not belong to a language package.

lost

This file is not a part of the language package itself.

Another file generated by **scripts/locales.pl**. If for any reason a translation does not appear anymore and can be deleted, it gets moved here. The last 50 or so entries deleted are saved here in case you made a typo, so that you don't have to translate everything again. If a tranlsation is missing, the lost file is checked first. If you maintain a language package, you might want to keep this safe somewhere.

more/all

This subdir and file is not a part of the language package itself.

If the directory more exists and contains a file called all it will be parsed in addition to the mandatory all (see above). The file is useful if you want to change some translations for the current installation without conflicting further upgrades. The file is not autogenerated and has the same format as the all, but needs another key (more_texts). See the german translation for an example or copy the following code:

```
#!/usr/bin/perl
# -*- coding: utf-8; -*-
# vim: fenc=utf-8

use utf8;
```

```
# These are additional texts for custom translations.
# The format is the same as for the normal file all, only
# with another key (more_texts instead of texts).
# The file has the form of 'english text'  => 'foreign text',

$self->{more_texts} = {

  'Ship via'                    => 'Terms of delivery',
  'Shipping Point'              => 'Delivery time',
}
```

4.7. Die kivitendo-Test-Suite

4.7.1. Einführung

kivitendo enthält eine Suite für automatisierte Tests. Sie basiert auf dem Standard-Perl-Modul `Test::More`.

Die grundlegenden Fakten sind:

- Alle Tests liegen im Unterverzeichnis `t/`.
- Ein Script (bzw. ein Test) in `t/` enthält einen oder mehrere Testfälle.
- Alle Dateinamen von Tests enden auf `.t`. Es sind selbstständig ausführbare Perl-Scripte.
- Die Test-Suite besteht aus der Gesamtheit aller Tests, sprich aller Scripte in `t/`, deren Dateiname auf `.t` endet.

4.7.2. Voraussetzungen

Für die Ausführung werden neben den für kivitendo eh schon benötigten Module noch weitere Perl-Module benötigt. Diese sind:

- `Test::Deep` (Debian-Paketname: `libtest-deep-perl`; Fedora: `perl-Test-Deep`; openSUSE: `perl-Test-Deep`)
- `Test::Exception` (Debian-Paketname: `libtest-exception-perl`; Fedora: `perl-Test-Exception`; openSUSE: `perl-Test-Exception`)
- `Test::Output` (Debian-Paketname: `libtest-output-perl`; Fedora: `perl-Test-Output`; openSUSE: `perl-Test-Output`)
- `Test::Harness` 3.0.0 oder höher. Dieses Modul ist ab Perl 5.10.1 Bestandteil der Perl-Distribution und kann für frühere Versionen aus dem CPAN[4] bezogen werden.
- `LWP::Simple` aus dem Paket `libwww-perl` (Debian-Panetname: `libwww-perl`; Fedora: `perl-libwww-perl`; openSUSE: `perl-libwww-perl`)
- `URI::Find` (Debian-Panetname: `liburi-find-perl`; Fedora: `perl-URI-Find`; openSUSE: `perl-URI-Find`)
- `Sys::CPU` (Debian-Panetname: `libsys-cpu-perl`; Fedora und openSUSE: nicht vorhanden)
- `Thread::Pool::Simple` (Debian-Panetname: `libthread-pool-simple-perl`; Fedora und openSUSE: nicht vorhanden)

Weitere Voraussetzung ist, dass die Testsuite ihre eigene Datenbank anlegen kann, um Produktivdaten nicht zu gefährden. Dazu müssen in der Konfigurationsdatei im Abschnit `testing/database` Datenbankverbindungsparameter angegeben werden. Der hier angegebene Benutzer muss weiterhin das Recht haben, Datenbanken anzulegen und zu löschen.

[4] http://www.cpan.org

Der so angegebene Benutzer muss nicht zwingend über Super-User-Rechte verfügen. Allerdings gibt es einige Datenbank-Upgrades, die genau diese Rechte benötigen. Für den Fall kann man in diesem Konfigurationsabschnitt einen weiteren Benutzeraccount angeben, der dann über Super-User-Rechte verfügt, und mit dem die betroffenen Upgrades durchgeführt werden. In der Beispiel-Konfigurationsdatei finden Sie die benötigten Parameter.

4.7.3. Existierende Tests ausführen

Es gibt mehrere Möglichkeiten zum Ausführen der Tests: entweder, man lässt alle Tests auf einmal ausführen, oder man führt gezielt einzelne Scripte aus. Für beide Fälle gibt es das Helferscript `t/test.pl`.

Will man die komplette Test-Suite ausführen, so muss man einfach nur `t/test.pl` ohne weitere Parameter aus dem kivitendo-Basisverzeichnis heraus ausführen.

Um einzelne Test-Scripte auszuführen, übergibt man deren Namen an `t/test.pl`. Beispielsweise:

```
t/test.pl t/form/format_amount.t t/background_job/known_jobs.t
```

4.7.4. Bedeutung der verschiedenen Test-Scripte

Die Test-Suite umfasst Tests sowohl für Funktionen als auch für Programmierstil. Einige besonders zu erwähnende, weil auch während der Entwicklung nützliche Tests sind:

- `t/001compile.t` -- compiliert alle Quelldateien und bricht bei Fehlern sofort ab
- `t/002goodperl.t` -- überprüft alle Perl-Dateien auf Anwesenheit von '`use strict`'-Anweisungen
- `t/003safesys.t` -- überprüft Aufrufe von `system()` und `exec()` auf Gültigkeit
- `t/005no_tabs.t` -- überprüft, ob Dateien Tab-Zeichen enthalten
- `t/006spelling.t` -- sucht nach häufigen Rechtschreibfehlern
- `t/011pod.t` -- überprüft die Syntax von Dokumentation im POD-Format auf Gültigkeit

Weitere Test-Scripte überprüfen primär die Funktionsweise einzelner Funktionen und Module.

4.7.5. Neue Test-Scripte erstellen

Es wird sehr gern gesehen, wenn neue Funktionalität auch gleich mit einem Test-Script abgesichert wird. Auch bestehende Funktion darf und soll ausdrücklich nachträglich mit Test-Scripten abgesichert werden.

4.7.5.1. Ideen für neue Test-Scripte, die keine konkreten Funktionen testen

Ideen, die abgesehen von Funktionen noch nicht umgesetzt wurden:

- Überprüfung auf fehlende symbolische Links
- Suche nach Nicht-ASCII-Zeichen in Perl-Code-Dateien (mit gewissen Einschränkungen wie das Erlauben von deutschen Umlauten)
- Test auf DOS-Zeilenenden (\r\n anstelle von nur \n)
- Überprüfung auf Leerzeichen am Ende von Zeilen
- Test, ob alle zu übersetzenden Strings in `locale/de/all` vorhanden sind
- Test, ob alle Webseiten-Templates in `templates/webpages` mit vom Perl-Modul `Template` compiliert werden können

4.7.5.2. Konvention für Verzeichnis- und Dateinamen

Es gibt momentan eine wenige Richtlinien, wie Test-Scripte zu benennen sind. Bitte die folgenden Punkte als Richtlinie betrachten und ihnen soweit es geht folgen:

- Die Dateiendung muss `.t` lauten.
- Namen sind englisch, komplett klein geschrieben und einzelne Wörter mit Unterstrichten getrennt (beispielsweise `bad_function_params.t`).
- Unterverzeichnisse sollten grob nach dem Themenbereich benannt sein, mit dem sich die Scripte darin befassen (beispielsweise `background_jobs` für Tests rund um Hintergrund-Jobs).
- Test-Scripte sollten einen überschaubaren Bereich von Funktionalität testen, der logisch zusammenhängend ist (z.B. nur Tests für eine einzelne Funktion in einem Modul). Lieber mehrere Test-Scripte schreiben.

4.7.5.3. Minimales Skelett für eigene Scripte

Der folgenden Programmcode enthält das kleinstmögliche Testscript und kann als Ausgangspunkt für eigene Tests verwendet werden:

```
use Test::More tests => 0;

use lib 't';

use Support::TestSetup;

Support::TestSetup::login();
```

Wird eine vollständig initialisierte kivitendo-Umgebung benötigt (Stichwort: alle globalen Variablen wie `$::auth`, `$::form` oder `$::lxdebug`), so muss in der Konfigurationsdatei `config/kivitendo.conf` im Abschnitt `testing.login` ein gültiger Login-Name eingetragen sein. Dieser wird für die Datenbankverbindung benötigt.

Wir keine vollständig initialisierte Umgebung benötigt, so kann die letzte Zeile

```
Support::TestSetup::login();
```

weggelassen werden, was die Ausführungszeit des Scripts leicht verringert.

4.8. Stil-Richtlinien

Die folgenden Regeln haben das Ziel, den Code möglichst gut les- und wartbar zu machen. Dazu gehört zum Einen, dass der Code einheitlich eingerückt ist, aber auch, dass Mehrdeutigkeit so weit es geht vermieden wird (Stichworte "Klammern" oder "Hash-Keys").

Diese Regeln sind keine Schikane sondern erleichtern allen das Leben!

Jeder, der einen Patch schickt, sollte seinen Code vorher überprüfen. Einige der Regeln lassen sich automatisch überprüfen, andere nicht.

1. Es werden keine echten Tabs sondern Leerzeichen verwendet.
2. Die Einrückung beträgt zwei Leerzeichen. Beispiel:

```
foreach my $row (@data) {
  if ($flag) {
    # do something with $row
  }

  if ($use_modules) {
    $row->{modules} = MODULE->retrieve(
      id   => $row->{id},
      date => $use_now ? localtime() : $row->{time},
    );
```

```
  }

  $report->add($row);
}
```

3. Öffnende geschweifte Klammern befinden sich auf der gleichen Zeile wie der letzte Befehl. Beispiele:

```
sub debug {
  ...
}
```

oder

```
if ($form->{item_rows} > 0) {
  ...
}
```

4. Schließende geschweifte Klammern sind so weit eingerückt wie der Befehl / die öffnende schließende Klammer, die den Block gestartet hat, und nicht auf der Ebene des Inhalts. Die gleichen Beispiele wie bei 3. gelten.

5. Die Wörter "`else`", "`elsif`", "`while`" befinden sich auf der gleichen Zeile wie schließende geschweifte Klammern. Beispiele:

```
if ($form->{sum} > 1000) {
  ...
} elsif ($form->{sum} > 0) {
  ...
} else {
  ...
}

do {
  ...
} until ($a > 0);
```

6. Parameter von Funktionsaufrufen müssen mit runden Klammern versehen werden. Davon nicht betroffen sind interne Perl-Funktionen, und grep-ähnliche Operatoren. Beispiel:

```
$main::lxdebug->message("Could not find file.");
%options = map { $_ => 1 } grep { !/^#/ } @config_file;
```

7. Verschiedene Klammern, Ihre Ausdrücke und Leerzeichen:

   Generell gilt: Hashkeys und Arrayindices sollten nicht durch Leerzeichen abgesetzt werden. Logische Klammerungen ebensowenig, Blöcke schon. Beispiel:

```
if (($form->{debug} == 1) && ($form->{sum} - 100 < 0)) {
  ...
}

$array[$i + 1]             = 4;
$form->{sum}              += $form->{"row_$i"};
$form->{ $form->{index} } += 1;

map { $form->{sum} += $form->{"row_$_"} } 1..$rowcount;
```

8. Mehrzeilige Befehle

   a. Werden die Parameter eines Funktionsaufrufes auf mehrere Zeilen aufgeteilt, so sollten diese bis zu der Spalte eingerückt werden, in der die ersten Funktionsparameter in der ersten Zeile stehen. Beispiel:

```
$sth = $dbh->prepare("SELECT * FROM some_table WHERE col = ?",
                     $form->{some_col_value});
```

b. Ein Spezialfall ist der ternäre Operator "?:", der am besten in einer übersichtlichen Tabellenstruktur organisiert wird. Beispiel:

```
my $rowcount = $form->{"row_$i"} ? $i
             : $form->{oldcount} ? $form->{oldcount} + 1
             :                     $form->{rowcount} - $form->{rowbase};
```

9. Kommentare

a. Kommentare, die alleine in einer Zeile stehen, sollten soweit wie der Code eingerückt sein.

b. Seitliche hängende Kommentare sollten einheitlich formatiert werden.

c. Sämtliche Kommentare und Sonstiges im Quellcode ist bitte auf Englisch zu verfassen. So wie ich keine Lust habe, französischen Quelltext zu lesen, sollte auch der kivitendo Quelltext für nicht-Deutschsprachige lesbar sein. Beispiel:

```
my $found = 0;
while (1) {
  last if $found;

  # complicated check
  $found = 1 if //
}

$i  = 0        # initialize $i
$n  = $i;      # save $i
$i *= $const;  # do something crazy
$i  = $n;      # recover $i
```

10. Hashkeys sollten nur in Anführungszeichen stehen, wenn die Interpolation gewünscht ist. Beispiel:

```
$form->{sum}      = 0;
$form->{"row_$i"} = $form->{"row_$i"} - 5;
$some_hash{42}    = 54;
```

11. Die maximale Zeilenlänge ist nicht beschränkt. Zeilenlängen unterhalb von 79 Zeichen helfen unter bestimmten Bedingungen, aber wenn die Lesbarkeit unter kurzen Zeilen leidet (wie zum Biespiel in grossen Tabellen), dann ist Lesbarkeit vorzuziehen.

Als Beispiel sei die Funktion `print_options` aus `bin/mozilla/io.pl` angeführt.

12. Trailing Whitespace, d.h. Leerzeichen am Ende von Zeilen sind unerwünscht. Sie führen zu unnötigen Whitespaceänderungen, die diffs verfälschen.

Emacs und vim haben beide recht einfache Methoden zur Entfernung von trailing whitespace. Emacs kennt das Kommande **nuke-trailing-whitespace**, vim macht das gleiche manuell über `:%s/\s\+$//e` Mit `:au BufWritePre * :%s/\s\+$//e` wird das an Speichern gebunden.

13. Es wird kein **perltidy** verwendet.

In der Vergangenheit wurde versucht, **perltidy** zu verwenden, um einen einheitlichen Stil zu erlangen. Es hat sich aber gezeigt, dass **perltidy**s sehr eigenwilliges Verhalten, was Zeilenumbrüche angeht, oftmals gut formatierten Code zerstört. Für den Interessierten sind hier die **perltidy**-Optionen, die grob den beschriebenen Richtlinien entsprechen:

```
-syn -i=2 -nt -pt=2 -sbt=2 -ci=2 -ibc -hsc -noll -nsts -nsfs -asc -dsm
-aws -bbc -bbs -bbb -mbl=1 -nsob -ce -nbl -nsbl -cti=0 -bbt=0 -bar -l=79
-lp -vt=1 -vtc=1
```

14. `STDERR` ist tabu. Unkonditionale Debugmeldungen auch.

    kivitendo bietet mit dem Modul `LXDebug` einen brauchbaren Trace-/Debug-Mechanismus. Es gibt also keinen Grund, nach `STDERR` zu schreiben.

    Die `LXDebug`-Methode "`message`" nimmt als ersten Paramter außerdem eine Flagmaske, für die die Meldung angezeigt wird, wobei "0" immer angezeigt wird. Solche Meldungen sollten nicht eingecheckt werden und werden in den meisten Fällen auch vom Repository zurückgewiesen.

15. Alle neuen Module müssen use strict verwenden.

    `$form`, `$auth`, `$locale`, `$lxdebug` und `%myconfig` werden derzeit aus dem main package importiert (siehe Globale Variablen [99]. Alle anderen Konstrukte sollten lexikalisch lokal gehalten werden.

4.9. Design 4.0 Less/CSS Anpassen

4.9.1. Einführung

Das Design 4.0 ist mit Less aufgebaut. Less erweitert CSS um einige nützliche Funktionen. Mehr dazu siehe: Getting started | Less.js[5] (Englisch)

Die Less Dateien befinden sich unter: `css/design40/less`

4.9.2. Less kompilieren

Da der Browser keine Less Dateien versteht müssen diese zu CSS kompiliert werden. Der zu verwendende compiler heisst `lessc`. Unter Debian kann dieser installiert werden mit:

```
sudo apt install node-less
```

Zum Kompilieren der Dateien aus dem Verzeichnis der Less Dateien den folgenden Befehl ausführen:

```
lessc -x style.less ../style.css
```

Dies produziert das neue Stylesheet oder gibt einen Fehler aus.

4.9.3. Lokale Anpassungen (nicht Entwicklung)

Für lokale Anpassungen sollten die folgenden Dateien verwendet werden:

- custom-variables.less (überschreibt Variablen in variables.less)
- custom.less (überschreibt am Ende alle originalen kivitendo Less Inhalte)

Diese Dateien können leer sein, wenn Sie das originale kivitendo Design nicht anpassen möchten. Die originalen Custom-Dateien enthalten nur Kommentare und deaktivierten Beispielcode.

Nach dem Anpassen müssen die Less Dateien wie oben beschrieben neu kompiliert werden.

4.9.3.1. Beispiel für Anpassung von Farben

Siehe dazu die Beispieldatei unter: `css/design40/less/custom_less_variables-example.less`

[5] https://lesscss.org/

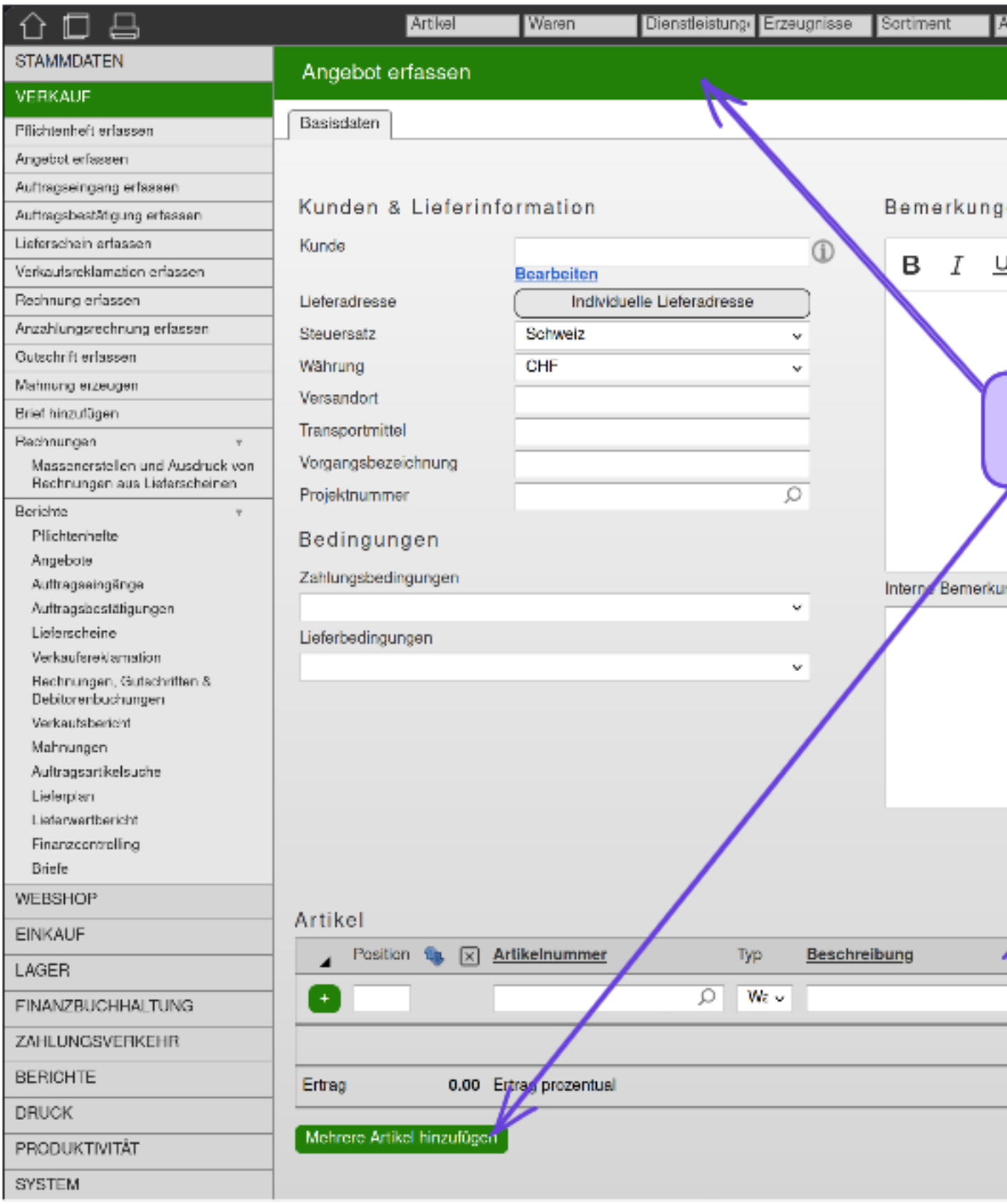


Abbildung 4.1. Screenshot mit einigen Variablennamen beschriftet

4.9.4. Weiteres

Weitere Informationen siehe auch: `css/design40/less/READ_ME.md`

HTML Entwickler Dokumentation kivitendo CSS (Design 4.0) unter: *Developer Tools → CSS Dokumentation* (`css/design40/manual/index.html`)

Tipp

Die HTML Seiten können auch als Live Test Seiten für das Design verwendet werden.

4.10. Dokumentation erstellen

4.10.1. Einführung

Diese Dokumentation ist in DocBook™ XML geschrieben. Zum Bearbeiten reicht grundsätzlich ein Text-Editor. Mehr Komfort bekommt man, wenn man einen dedizierten XML-fähigen Editor nutzt, der spezielle Unterstützung für DocBook™ mitbringt. Wir empfehlen dafür den XMLmind XML Editor[6], der bei nicht kommerzieller Nutzung kostenlos ist.

4.10.2. Benötigte Software

Bei DocBook™ ist Prinzip, dass ausschließlich die XML-Quelldatei bearbeitet wird. Aus dieser werden dann mit entsprechenden Stylesheets andere Formate wie PDF oder HTML erzeugt. Bei kivitendo übernimmt diese Aufgabe das Shell-Script **scripts/build_doc.sh**.

Das Script benötigt zur Konvertierung verschiedene Softwarekomponenten, die im normalen kivitendo-Betrieb nicht benötigt werden:

- Java[7] in einer halbwegs aktuellen Version
- Das Java-Build-System Apache Ant[8]
- Das Dokumentations-System Dobudish für DocBook™ 4.5, eine Zusammenstellung diverser Stylesheets und Grafiken zur Konvertierung von DocBook™ XML in andere Formate. Das Paket, das benötigt wird, ist zum Zeitpunkt der Dokumentationserstellung `dobudish-nojre-1.1.4.zip`, aus auf code.google.com[9] bereitsteht.

Apache Ant sowie ein dazu passendes Java Runtime Environment sind auf allen gängigen Plattformen verfügbar. Beispiel für Debian/Ubuntu:

```
apt-get install ant default-jre
```

Nach dem Download von Dobudish muss Dobudish im Unterverzeichnis `kivitendo-erp/doc/build` entpackt werden. Beispiel unter der Annahme, das Dobudish™ in `$HOME/Downloads` heruntergeladen wurde:

```
cd kivitendo-erp/doc/build
unzip $HOME/Downloads/dobudish-nojre-1.1.4.zip
```

Damit auf Dobudish zugegriffen werden kann, muss ein Link mit dem Namen `dobudish` erstellt werden, der auf den Ordner `dobudish-nojre-1.1.4` zeigt.

```
cd kivitendo-erp/doc/build
ln -sf dobudish-1.1.4 dobudish
```

[6] http://www.xmlmind.com/xmleditor/
[7] http://www.oracle.com/technetwork/java/index.html
[8] http://ant.apache.org/
[9] https://code.google.com/archive/p/dobudish/downloads

4.10.3. PDFs und HTML-Seiten erstellen

Die eigentliche Konvertierung erfolgt nach Installation der benötigten Software mit einem einfachen Aufruf direkt aus dem kivitendo-Installationsverzeichnis heraus:

```
./scripts/build_doc.sh
```

4.10.4. Einchecken in das Git-Repository

Sowohl die XML-Datei als auch die erzeugten PDF- und HTML-Dateien sind Bestandteil des Git-Repositories. Daraus folgt, dass nach Änderungen am XML die PDF- und HTML-Dokumente ebenfalls gebaut und alles zusammen in einem Commit eingecheckt werden sollten.

Die "`dobudish`"-Verzeichnisse bzw. symbolischen Links gehören hingegen nicht in das Repository.