Accounting

This feature has not really evolved much between Tiki7 and Tiki19. Benoit G., Victor and Marc will revamp for Tiki21 or Tiki22.

Topics & Developments related to accounting in Tiki:

Notes of Accounting in Tiki7

- There should be links in the menus to tiki-accounting.php (cdrwhite: when it is ready)
- tiki-accounting_entry.php -> Fatal error: Call to undefined method DateTime::getTimestamp() in lib/accounting/accountinglib.php on line 528 (cdrwhite: switched to Zend_Date)
- tiki-accounting_books.php needs a permission check (cdrwhite: please elaborate, checking for tiki_p_acct_create_book to create a new book and the list of books is filtered by Perms::filter)

1.2. TimeSheet and billing

TimeSheet

1.3. Internal Tiki Association Accountancy

This will be a simple use case: Dogfood

Here is an example of how to add entries and have basic calculations
http://profiles.tiki.org/Time_Sheet

marc laporte wrote:

> *For advanced calculations, we'll use the spreadsheet* 🙂

lindon wrote:

> *Tiki payment also seems to have invoices of sorts and deals with money coming in - so I was thinking it might be good to start there and create some financial reporting around revenues and donations received through payment. Second main part is dealing with money going out - expenses and donations paid. Beyond that it's how far we want to take it - i could see tracking and reconciling cash, perhaps budgeting. Not sure we're looking for a full double entry accounting system that can generate balance sheets etc, but would like to know what others are thinking.*

Ideas for the Accounting Feature

User Interface and Functionality

Thinking there should be one main page with tabs across the top for the main areas. Probably the following tabs/areas:

- Dashboard
- Invoices

- Receipts
- Payments
- Reports

Functionality

Users should be able to:

- Set up multiple sets of books
- Choose currency
- Have payments from payment feature go directly into invoices and receipts
- Add, edit, delete items
- Categorize receipts and payments into income/expense/balance sheet categories
- Filter/sort items by date (or range), counterparty, category, status, amount, description
- Import/export data
- Generate reports of
    - Overview of receipts and payments, with choice of date ranges
    - Reports of net earnings and cash flows for choice of periods
    - Open, closed, overdue invoices
    - Payments/receipts by counterparty, category, status

Admin and Preferences

User will need to be able to set:

- Currency
- Amount format (thousands and decimal separator)
- Date format
- Fiscal year end
- Whether they want direct feed from payment feature (including pre-existing transactions)
- Categories
- Taxes (?)

Development Plan

Current plan is to base this feature on trackers, permissions and categories as described below:

Two trackers would be automatically set up when the accounting feature is enabled: Transactions and Counterparties. Transactions is the main tracker with Counterparties used to provide a dropdown list of counterparties for the counterparty field in the Transcations tracker (so the Counterparty tracker is not absolutely necessary). The Transactions tracker would have the following fields:

Field	Type	Comments
Id	auto-increment	auto transaction number
Entry	numeric	for double-entry accounting so that 2 or more tracker items that are part of the same entry would have the same entry number

Field	Type	Comments
Date	jscalendar	date of transaction
Counterparty	item link	
Amount	currency amount	enter either a positive or negative number. Don't use debit/credit terminology.
Account	category	
Description	text field	
Long description	textarea	not absolutely necessary

Permissions would be set for each field with one group (Acctng-view) with rights to see the items and the other (Acctng-edit) with editing rights.

The account structure would be set through categories. The first two levels would be preset as follows:

Top Level	Level 1	Level 2, etc.
Accounting	Assets	Asset accounts defined by user
	Liabilities	Liability accounts defined by user
	Revenues	Revenue accounts defined by user
	Expenses	Expense accounts defined by user
	Equity	Equity accounts defined by user

Some of the next tasks would be:

- Tracker form needs to refresh so that many transactions can be added easily.
- To produce a balance sheet or income statement by account, enhance Plugin Trackerlist or create new plugin to pull totals by category (account) which could be displayed with or without the detail items (toggle using jquery)
- To have the accounts show in the right order on the balance sheet or income statement, enhance categories to be allow the user to set display order (alphabetical won't work too well for financial statements)
- Ensure date filtering for date ranges works or enhance to do so
- If double-entry accounting will be supported, then enhance input form to allow user to add additional items with the same entry number as the first item while showing a running total showing whether the entry balances or not
- Link payment to revenues, perhaps by adding a parameter for the account to the appropriate payment plugins. Combination of that and new preference set to link payment to accounting would cause an entry to be made when a payment is entered.

- Decide whether users will be able to "close" the books. Benefit is better speed over time as each balance sheet or income statement won't need to be built up from all cumulative transactions for each account every time. Disadvantage is that it would be complex to code and would probably involve adding additional database tables or trackers.

1.4. Feature: Tiki Payment

Introduced in Tiki5, see doc: Payment

Marc Laporte: We have Tiki payment which lets you handle membership and the shopping cart. But it's not really accounting. I think we could do book keeping with trackers though

1.5. Feature: Community Currencies (CC-take2)

cc-take2 project aims to:

- allow users within a tiki site trade with community currency (which is not euros, nor dollars, nor conventional money but community agreed local money, such as "hours" or any other locally created and managed currency) within a barter/exchange network.
- allow the other users to see the payments/transactions of other community members, etc., so that some public financial reporting around payments and transactions would be needed to allow everybody see clearly where the community-currency money comes from and where it goes.
  Some work was started by mose and others years ago with the Mod CC, which had the basic features by then (2005 or so), but it's currently a little bit outdated compared to Tiki stable releases.
  - http://cc.tiki.org/Currencies+Service+Draft
  - http://mods.tiki.org/details.php?type=features&mod=cc

In order to allow payments with other sources than PayPal (the only one implemented in Tiki5, afaik, so far), we have to decide whether to:

- (a) implement (or use whatever is in Tiki5 already with the payment feature, if feasible) some accounting system within Tiki itself. Maybe reusing and extending the work done by mose & co. in the CC Mod?
- (b) use some specific third party software for such task like CClite (jonny already have that (nearly?) installed on http://c2c.ourproject.org/cgi-bin/cclite.cgi
  And as far as I know, mose already coded some sort of communication between CC mod and a former version of cclite, so it's worth having a look at mose's code...
- (c) allow users to pay through (a) or (b)

1.6. Feature: Mod tinvoice

http://mods.tiki.org/details.php?type=features&mod=tinvoice

Adaption of CCSG accounting for tiki7

cdrwhite set up a simple accounting system for the Computer-Club Siebengebirge e.V. in 2002. When the intranet was converted from the old code to tiki 1.x, the accounting was integrated but never actually

committed to trunk for a couple of reasons:

- most of the code was not well structured
- table/field names and templates which came from the old system were in German only, there was no support for translation
- the system was closely tied into the ccsg membership management

Status

Currently this system is being integrated into tiki.

So far, the following functionality already exists:

- the database structure is almost finished
- A well documented core accounting lib contains all base functions for managing accounts and do the actual booking
- Pages/Templates exist for booking and account management

The following features need polishing:

- External Bank Account statements (CSV/Text) can be imported as a "to do list":
    - The Import Routine must become more flexible to handle different formats
    - Validation detects overlapping statements/double imports
    - Template recognition for recurring transactions e.g. Electricity bill every month the same amount, going to the same target account
    - Javascript stuff should be converted to jquery
- Cancellation needs polishing
- Multiple books

The following features are missing:

- Automated handling of VAT
- Handling of model charts of account (SKR03, SKR04 etc)
- Flexible report/export system to get data out
    - Report for a flexible time period
    - Report for an account
    - Tax reports
- Handling templates for recurring transactions
- Close a book for a year with closing statements and generate a new book for the following year

Timeline

A working alpha version has been committed to trunk (r30948), the missing parts will be appended bit by bit.

Database structure

tiki_acct_book - Defining books for different periods or clients

tiki_acct_book

```
CREATE TABLE `tiki_acct_book` (
  `bookId` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `bookName` varchar(255) NOT NULL,
  `bookClosed` enum('y','n') NOT NULL DEFAULT 'n',
  `bookStartDate` date NOT NULL,
  `bookEndDate` date NOT NULL,
  `bookCurrency` varchar(3) NOT NULL DEFAULT 'EUR',
  `bookCurrencyPos` int(11) NOT NULL,
  `bookDecimals` int(11) NOT NULL DEFAULT '2',
  `bookDecPoint` varchar(1) NOT NULL DEFAULT ',',
  `bookThousand` varchar(1) NOT NULL DEFAULT '.',
  `exportSeparator` varchar(4) NOT NULL DEFAULT ';',
  `exportEOL` varchar(4) NOT NULL DEFAULT 'LF',
  `exportQuote` varchar(4) NOT NULL DEFAULT '"',
  `bookAutoTax` enum('y','n') NOT NULL DEFAULT 'y',
  PRIMARY KEY (`bookId`)
);
```

This table contains the core definition of a book, its name, start and end date (only booking beetween those dates should be allowed, currently not enforced). When the book is marked as closed, transactions should no longer be possible but viewing the data and generating reports or exporting data should still be allowed.
Also some display settings which are specific for each book are set here: the currency and the position of the currency symbol, the number of decimals and the decimal point an thousands separator.
The export settings are the defaults for all CSV exports: The separator between fields, the EndOfLine (CR,LF or CRLF) and the Quote for text fields.
The last option is a flag which defines if automated tax deduction should be enabled (For example, if a client pays me 119 EUR to my bank Account, I have to book the bank account on debit side with 119 EUR and split credit into 100 EUR for the "profits" account and 19 EUR for the "VAT received" account. This will always be the same for the normal profits account, so the splitting can be done automagically.

tiki_acct_account - Defining accounts for each book

tiki_acct_account

```
CREATE TABLE `tiki_acct_account` (
  `accountBookId` int(10) unsigned NOT NULL,
  `accountId` int(10) unsigned NOT NULL DEFAULT '0',
  `accountName` varchar(255) NOT NULL,
  `accountNotes` text NOT NULL,
  `accountBudget` double NOT NULL DEFAULT '0',
  `accountLocked` int(1) NOT NULL DEFAULT '0',
  `accountTax` int(11) NOT NULL DEFAULT '0',
  `accountUserId` int(8) NOT NULL DEFAULT '0',
  PRIMARY KEY (`accountBookId`,`accountId`),
  KEY `accountTax` (`accountTax`)
);
```

Each account has a number. There are common numbering schemata for accounts for different purposes

(like SKR03 or SKR04 for companies in Germany). This unique number is also the accounts id. The account name should describe the accounts purpose, a longer text can be stored in the notes field. If there is a budget planned for this account, it can be stored as well, allowing to see if the business plan for a year is working out as well. Accounts which have been already used can't be deleted as they are still needed for reference. Declaring them as "locked" will prevent usage in the booking mask. The tax automation rule associated with the account can be defined here as well. An account can also be associated with a user id, allowing a user who has no other rights in the current book to view his own personal account (we use this in our membership management, each member has one account, so it does not create confusion, if people pay their membership fees for a year in advance or quarterly or monthly).

tiki_acct_journal and tiki_acct_item - The transactions

tiki_acct_journal and tiki_acct_item

```
CREATE TABLE `tiki_acct_journal` (
  `journalBookId` int(10) unsigned NOT NULL,
  `journalId` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `journalDate` date NOT NULL DEFAULT '0000-00-00',
  `journalDescription` varchar(255) NOT NULL,
  `journalCancelled` int(1) NOT NULL DEFAULT '0',
  `journalTs` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`journalBookId`,`journalId`)
);

CREATE TABLE `tiki_acct_item` (
  `itemJournalId` int(10) unsigned NOT NULL DEFAULT '0',
  `itemAccountId` int(10) unsigned NOT NULL DEFAULT '0',
  `itemType` int(1) NOT NULL DEFAULT '-1',
  `itemAmount` double NOT NULL DEFAULT '0',
  `itemText` varchar(255) NOT NULL DEFAULT '',
  `itemTs` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`itemJournalId`,`itemAccountId`,`itemType`)
);
```

As one side of a statement (either credit or debit) can be split into several accounts, it is necessary to have two tables here. The Journal contains the data for the whole entry, which is mainly a unique id, the booking date, a text description for the whole process and if the entry has been cancelled.

For each entry in the journal, there must be at least 2 entries in the item table using the same journal id to create a valid entry. The item type is either -1 (debit) or 1 (credit). As both sides of an entry must always be equal, a simple query can be used to determine incomplete/defective entries by multiplying amount (which is always positive) by type and summing them up for each journal entry.

Especially when there are multiple entries on one side, the additional text can be used. For example, the amount transferred to the landlord usually consists of multiple parts, but is transferred in one amount from the bank account. So on credit side, there is only one account, the bank account but on debit side there are multiple accounts which are all in the same range when using a standard account schema. So we could state for example why in december we are transferring a different amount for gas or electricity than in the other months before which makes the journal easier to read later on.

One big problem with accounting is the topic of authenticity. Using timestamps, later alterations could be detected more easily (however a clever accountant would also manipulate those 😕)

tiki_acct_bankaccount - linking your bank with the accounting

tiki_acct_bankaccount

```
CREATE TABLE `tiki_acct_bankaccount` (
  `bankBookId` int(10) unsigned NOT NULL,
  `bankAccountId` int(10) unsigned NOT NULL,
  `externalNumber` int(10) NOT NULL,
  `bankCountry` varchar(2) NOT NULL,
  `bankCode` varchar(11) NOT NULL,
  `bankIBAN` varchar(63) NOT NULL,
  `bankBIC` varchar(63) NOT NULL,
  `bankDelimeter` varchar(15) NOT NULL DEFAULT ';',
  `bankDecPoint` varchar(1) NOT NULL DEFAULT ',',
  `bankThousand` varchar(1) NOT NULL DEFAULT '.',
  `bankHasHeader` tinyint(1) NOT NULL DEFAULT '1',
  `fieldNameAccount` varchar(63) NOT NULL,
  `fieldNameBookingDate` varchar(63) NOT NULL,
  `formatBookingDate` varchar(31) NOT NULL,
  `fieldNameValueDate` varchar(63) NOT NULL,
  `formatValueDate` varchar(31) NOT NULL,
  `fieldNameBookingText` varchar(63) NOT NULL,
  `fieldNameReason` varchar(63) NOT NULL,
  `fieldNameCounterpartName` varchar(63) NOT NULL,
  `fieldNameCounterpartAccount` varchar(63) NOT NULL,
  `fieldNameCounterpartBankCode` varchar(63) NOT NULL,
  `fieldNameAmount` varchar(63) NOT NULL,
  `amountType` int(10) unsigned NOT NULL,
  `fieldNameAmountSign` varchar(63) NOT NULL,
  `SignPositive` varchar(7) NOT NULL,
  `SignNegative` varchar(7) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`bankBookId`,`bankAccountId`)
);
```

This is actually the import specification for CSV/text bank statements. Generally speaking, if there is an external source triggering transactions on one account in the book, here the rules for importing data provided by the external "supplier" can be specified.

First of all, the internal book and account id are specified. The next fields, the external number (your bank account number, currently maximum of 10 digits), the country code, bank code and the international IBAN and BIC are specified for (future) exports of transactions (cdrwhite is dreaming of a dtaus generation tool or a link to finTS applications but lacks the time for that).
The next fields define the delimeter, decimal point, thousands separator of the source file. it also specifies, if the file provided by the bank has a header line with field names or not (cdrwhite: maybe this should be changed to int to allow multiple header lines, but I have not seen german banks so far which do that).
The field names define how your bank is labelling the different columns. Not all fields must be present at all times but at least one date (value preferred over booking date) should be there and of course, the reason field is the easiest way of seeing what the sender of the money wanted to tell. The booking text usually defines the type of transaction like debit, credit, fees, salary, if provided at all. The counterpart data will be used in automagically assigning booking templates to well known transactions (like the recurring payment to the landlord or our 50+ monthly membership fees).

The amount field is also a very important field for automated processing of a bank statement, however 4

out of 5 german banks I work with have different ways of specifying the amount. The simplest way is having positive and negative amounts, some banks use two columns for positive and negative amounts (with and without -) and some use a separate column stating S for debit and H for credit. This is why we also need the amount type and sign fields. (todo: fix inconsistent capitalization)

tiki_acct_statement - Data imported from the bank statement

tiki_acct_statement

```
CREATE TABLE IF NOT EXISTS `tiki_acct_statement` (
  `statementBookId` int(10) unsigned NOT NULL,
  `statementAccountId` int(10) unsigned NOT NULL DEFAULT '0',
  `statementId` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `statementBookingDate` date NOT NULL,
  `statementValueDate` date NOT NULL,
  `statementBookingText` varchar(255) NOT NULL,
  `statementReason` varchar(255) NOT NULL,
  `statementCounterpart` varchar(63) NOT NULL,
  `statementCounterpartAccount` varchar(63) NOT NULL,
  `statementCounterpartBankCode` varchar(63) NOT NULL,
  `statementAmount` double NOT NULL,
  `statementJournalId` int(10) unsigned NOT NULL DEFAULT '0',
  `statementStackId` int(11) NOT NULL,
  PRIMARY KEY (`statementBookId`,`statementAccountId`,`statementId`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;
```

This contains the data from all bank statements after they have been imported. The moment, a statement is converted into an entry in the stack, the stackId is set to the id of the entry in the stack and as soon as it goes to the journal, the journalId is set to the journalId. This allows easy referencing of the bank statements with the actual bookings. In this table all fields which make sense for import are defined with field names and field order defined by us 🥲

tiki_acct_stack and tiki_acct_stackitem - the sandbox

tiki_acct_stack and tiki_acct_stack_item

```
CREATE TABLE `tiki_acct_stack` (
  `stackBookId` int(10) unsigned NOT NULL,
  `stackId` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `stackDate` date NOT NULL DEFAULT '0000-00-00',
  `stackDescription` varchar(255) NOT NULL,
  `stackTs` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`stackBookId`,`stackId`)
);

CREATE TABLE `tiki_acct_stackitem` (
  `stackItemStackId` int(10) unsigned NOT NULL DEFAULT '0',
  `stackItemAccountId` int(10) unsigned NOT NULL DEFAULT '0',
  `stackItemType` int(1) NOT NULL DEFAULT '-1',
  `stackItemAmount` double NOT NULL DEFAULT '0',
  `stackItemText` varchar(255) NOT NULL DEFAULT '',
  PRIMARY KEY (`stackItemStackId`,`stackItemAccountId`,`stackItemType`)
);
```

The stack is a kind of journal "light" where transactions can be altered or deleted. Only when they are

finally approved, they go to the journal and can't be changed any more. The data structure is almost the same as for the journal/item tables, only the timestamps and "Cancelled" fields are missing, as they are not used.

By splitting the rights to book into the stack and into the real journal, a 4-eyes-policy can be established or less qualified people can do the stupid typing stuff and the accountant just has to check/correct their data before promoting stuff to the journal.

tiki_acct_tax - Tax types

tiki_acct_tax

```
CREATE TABLE `tiki_acct_tax` (
  `taxBookId` int(10) unsigned NOT NULL,
  `taxId` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `taxText` varchar(63) NOT NULL,
  `taxAmount` double NOT NULL DEFAULT '0',
  `taxIsFix` enum('y','n') NOT NULL DEFAULT 'n',
  PRIMARY KEY (`taxBookId`,`taxId`)
);
```

This table defines the different types of taxes for automated splitting. The default type 0 (no tax relevance) does not have to be defined.
Taxes can be specified individually for each book and globally (for all books of the site by setting taxBookId to 0). The tax amount is usually a factor/percentage, but it might also be possible to have a fixed amount (Which would be funny, having a fixed tax of 1 EUR, even if the amount is only 1 Cent).

This allows me to specify the VAT I have to pay (Vorsteuer, which I can later deduct from the VAT i collected from my customers), where we actually have 2 of them 7% and 19% (plus of course some special stuff for inter european building contracts and so on) and also the VAT i have to collect from my customers (Umsatzsteuer). Each entry has a unique Id and can be referenced by the appropriate accounts.

Pages/files so far

lib/accounting/accountinglib.php

Contains the core functionality (more text to follow, but its documented in the code. Really!)

tiki-accounting_books.php

Create books and view the books I have access to.

tiki-accounting.php

This is the main page for each book, here you can see a list of accounts and trigger various actions. This will become the main dashboard for an accountant when working in one book.

tiki-accounting_account.php

Manage the actual accounts

tiki-accounting_entry.php

Actually does the booking of a statement

tiki-accounting_export.php

This is a general export routine (so far only CSV, but later on other formats like print(html) or jquery.Sheet should be handled here

tiki-accounting_stack.php

Allowing the stack as "sandbox" or to implement a 4-eyes-policy

Permissions

tiki_p_acct_create_book

This permission should be assigned globally to alkl accountants who can create close a book

tiki_p_acct_manage_accounts

This permission should be assigned as object permission on each book. The people in the corresponding group can create/edit/lock accounts and even delete them if they are unused

tiki_p_acct_book

This (object) permission allows creating actual entries in the journal or transfer entries from the stack to the journal

tiki_p_acct_book_stack

This object permission only allows creation of new entries in the stack, where statements can be changed. They only become valid statements after being confirmed (moved to the journal by someone with tiki_p_acct_book

tiki_p_acct_book_import

This object permission allows importing statements from external accounts as CSV/text files

tiki_p_acct_manage_template

This object permission allows creating and editing templates for typical transactions which are recurring but a lot of work to type in

Related links

- Shopping Cart
- Credits
- Payment
- Payments Pro
- PayPal Subscription
- Community Currencies

- http://www.ohloh.net/tags/erp
- http://ck-erp.org/tikiwiki/ck-ledger/index.php
- http://www.sql-ledger.com/
- http://www.dolibarr.org/onlinedemo
- http://phpbms.org/trial/
- webERP
    - Using webERP with a Wiki
    - https://www.ohloh.net/forums/10/topics/8009
- http://www.simpleinvoices.org/
- https://www.ohloh.net/tags/accounting
- https://www.ohloh.net/tags/erp
- http://openexchangerates.org/
- http://www.allthemob.com.au/tiki-index.php?page=ERP
- http://frontaccounting.com/
- http://www.phpcompta.be/NOALYSS
- OpenConcerto
- https://www.openhub.net/p/gnucash (desktop app)
- https://kmymoney.org/ (desktop app)
- http://imperium.edoceo.com (and has a list of others)

ERP discussion copied from Suite Brainstorming

ERP/accounting is a critical business function. Here are several contenders:

- Compiere Strengths: Compiere is an industrial strength ERP solution. It is a very mature, feature rich, and flexible application. Weaknesses: It is controlled/supported by a single company. It is released under a dual (open source/commercial) license. Compiere is java based, and uses either Oracle or Postgres backends.
- Openbravo is a fork of Compiere. It has similar strengths and weaknesses compared to Compiere (powerful & mature, controlled by a single company & dual licenses).
- ADempiere is also a fork of Compiere, but it is supported by a strong development community with a democratic community process. There is a great article about the advantages of loosely coupling Adempiere via an ESB here.
- WebERP is a Apache/MySQL/PHP app that is very mature and well supported. It has fewer bells and whistles than Compiere and it's forks, but is still quite comprehensive. It has support for being loosly connected to a Wiki (Wacko Wiki). The out of the box wiki support is for wiki pages, not a tracker type CRM app.
- SQL-Ledger is a perl/postgres based app. It was forked because of issues with the main developer (notice the "dot com" domain name).
- Ledger SMB is the sql-ledger fork. There are a few heavyweight developers behind this project, but it moves very slowly...
- http://www.xtuple.com/postbooks
- https://cwiki.apache.org/confluence/display/OFBIZ/Apache+OFBiz+User+List -> Open for Business has some derivatives
- http://webzash.org
- https://github.com/rezzza/accounting

https://www.ohloh.net/tags/erp/php
Activity level comparison

https://www.openhub.net/p/odoo is likely the most active

Akaunting

https://packagist.org/packages/akaunting/
https://github.com/akaunting/akaunting/blob/master/composer.json

- Looking for Chart of Accounts, Journal Entry, General Ledger, Balance Sheet, and Trial Balance? This app brings the double-entry accounting features to Akaunting. Akaunting ships with single-entry accounting feature set, same as FreshBooks. However, some of the businesses, whether they use the cash-basis accounting method or the accrual accounting method, use double-entry bookkeeping to keep their books. Double-entry accounting is a practice that helps minimize errors and increases the chance that your books balance.

alias

- ERP