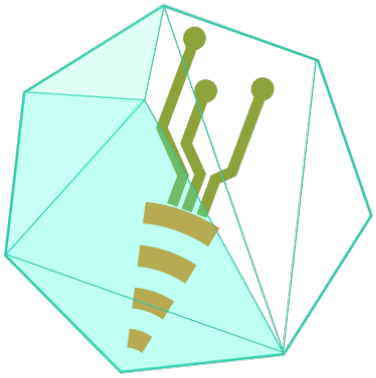
rechnung

Feb 16, 2021

Contents:

rechnung is a file based tool to manage invoices and contracts. It's designed for tech people, who'd like to generate bullet proof invoices without leaving their well known environment: the terminal.

rechnung provides easy-to-use batch functions and an intuitive CLI to all functionality.To allow for even the craziest modifications, all generated content, can be reedited by yourself, scripts or whatever you like, as everything is stored in files.



CHAPTER 1

Getting started

1.1 Design

This document describes the design thoughts and specification of *rechnung* - a command line based contract and invoice management.

The state of the tool is exclusively stored in files. This distinguishes *rechnung* from other tools on the market (e.g. invoice ninja). Saving all information and state in files has various advantages. It can be tracked with git or any other revision control system. Reverting to a previous state of the tool is not problematic. The user can use his known skills of editing files, to adjust the state, sothe tool can be used in cornercases than we can't even think of. Revision of the data directory should save you from trouble with your local finance authority (depends in which country you are in).

1.1.1 Workflow

This tool is designed to assist you, if you have the following worklow.

You create a new contract for a customer. A contract contains information about the customer, product (and addons), as well as contact information. The contract is sent to the customer by email (using the *send contract* command and to be signed and returned.

The contract contains a *product* section. Based on the description the matching product description document from the assets folder is attached to the contracts email as well as the terms and/or returns policy.

The signed contract (as jpg or pdf file) is saved into the *signed_contracts* directory. After that, the customer can be imported into the system.

The *initial cost* for the product is appended to the file in the *billed_items* directory.

Every month (or any other cycle you wish) you run the *bill* command. Every active contract will get all his products appended to his billed items. This way, you keep track of what to put on the next invoice. You can of course manually add positions to the positions file.

If the time is come to send invoices, first you run the *create invoice* command. This will create an invoice with all positions which have not been put on an invoice so far. Also the invoice id is added to the positions for them to not be billed again.

You can now inspect the invoice files. If everything is correct, run the *render-all* command to render all invoices. You can inspect them again and can now use *send invoices* to bulk send the invoices to your customers.

Now it's time to check your bank account. You can create a match file in the *matchings* directory for every customer to match for the customers IBAN, name, transaction subject, etc. The *annotate* command will annotate a bank statement, i.e. it will try to match a customer for every incoming payment. If there is (only one matching customer) this transaction is added to the customers file in the *saldo* directory, where also all totals of invoices are listed. A saldo file can be created using the *create saldo* command, which can be send to the customer to remind them of due payments.

1.2 Installation

1.2.1 Prerequisites

rechnung is develop under Linux and currently only tested with Linux. If you happen to use it under Windows, Mac, ... please be aware that unexpected behaviour may be observed.

Except from having Linux as your operating System, **Python 3.7** is required. Unfortunately this is the only Python version currently supported, as one of the dependencies (html5lib-python, does not support Python >3.8) and we make use of dataclasses which is available for Python >3.7.

1.2.2 User Installation

Currently the recommended installation path is via this repository on github. To start, clone this repository to your machine:

```
$ git clone https://github.com/westnetz/rechnung
```

And install the package using *pip* or the provided *make install* method.

```
$ make install
```

1.3 Quickstart

This section is a quick walkthrough all the features.

1.3.1 Initialization

Before you can start generating your own invoices you need to setup your working directory for *rechnung*. By invoking

```
$ rechnung init
```

all required directory and configuration files will be placed in the current working directory. It is recommended to do this in a clean directory.

1.3.2 Configuration and Customization

You can now edit the *rechnung.config.yaml* file to your needs. You need to enter the credentials for the mail server if you want to send out your invoices by email.

Customization of the invoices can be done by editing the invoice template *templates/invoice_template.j2.html* and the corresponding stylesheet in *assets/inovice.css*.

1.3.3 Creating invoices

After creating your customers, you can create your first invoices. The following command

```
$ rechnung create-invoices 2019 10
```

will create invoices for all customers who have a contract starting before and ending after october 2019.

You can force overwrite of existing invoices by giving the *–force/-f* option

```
$ rechnung create-invoices --force 2019 10
```

Individual invoices can be created by giving a specific customer id (cid)

```
$ rechnung create-invoices -c 1000 2019 10
```

After creating your invoices you can doublecheck for correctness.

1.3.4 Rendering invoices (create pdf files)

If everything is correct, you are ready to create your pdf invoice documents.

```
$ rechnung render-all
```

This command will render all invoice yaml files, which have no corresponding pdf file. I.e. if you happen to spot an error in an invoice pdf. Simply delete the pdf file, correct the mistake in the invoice yaml, and run the command again.

1.3.5 Sending invoices

If you want to use the included mail delivery service, you should customize the invoice mail template to your needs: *assets/invoice_mail_template.j2*.

After doing that, you can send all the invoices you just created to your customers:

```
$ rechnung send 2019 09
```

This command will send all invoices with the given suffix to the customer given in the invoice yaml file.

And that's it!

1.4 rechnung

1.4.1 rechnung package

Subpackages

rechnung.tests package

Submodules

rechnung.tests.test_cli module

Module contents

Submodules

rechnung.cli module

rechnung.contract module

rechnung.contract.**get_contracts**(*settings*, *year=None*, *month=None*, *cid_only=None*, *inactive=False*)
Fetches all contracts from the settings.contracts_dir directory. Returns a dict with all active contracts, i.e. contracts with started in the past.

rechnung.contract.**render_contracts**(*settings*)
Renders all contracts as pdfs to settings.contracts_dir

rechnung.contract.**send_contract**(*settings*, *cid*)
Sends the contract specified with the cid via email to the customer.

If set, the policy and the product description of the main product will be attached.

rechnung.helpers module

rechnung.helpers.**exit_on_miss**(*key*, *key_name*, *required_keys*)
Checks existence of the given key in the given list. If the key is not found in the list, an error message is displayed, and the program is exited with exit code 1.

Args: key: the key to be found key_name: a verbose name of the key, e.g. the variable name required_keys: the list where the key is to be found

rechnung.helpers.**generate_email**(*settings*, *mail_to: str*, *mail_subject: str*, *mail_text: str*, *files=None*)
Generate EmailMessage

Args: settings mail_to: receiver mail address mail_subject: mail subject mail_text: mail text files (touple): list of file_path and file_name

Returns: email.EmailMessage

rechnung.helpers.**generate_pdf**(*html_data*, *css_data*, *path*)
Takes rendered HTML template and filename and converts it to a PDF invoice using weasyprint.

Args: rendered (str): Rendered invoice HTML invoice_path: Complete path where the invoice will be written to

rechnung.helpers.**generate_yaml**(*object*, *filename*)
Small wrapper around the yaml dump function.

Args: object: Python object to be stored. filename: Filename of the yaml file.

rechnung.helpers.**get_template**(*template_filename*)
Takes the path to the jinja2 template and returns a jinja2 Template instance with the contents of the file.

Args: template_filename (str): full path to the template file (jinja2)

Returns: Template: jinja2 Template instance.

rechnung.helpers.**read_with_default**(*prompt*, *default=None*)
Prompts the user to enter some value. If a default value is given, the function will return that value, if the user presses enter, or enters only characters which are stripped by strip().

Args:

prompt: The prompt to be displayed to the user.

default: The default value to be returned if user enters nothing default: None

Returns:

str(): the default if set, empty string if not default and no input, the input if something was entered.

rechnung.helpers.**send_email**(*msg*, *server*, *username*, *password*, *insecure=True*)
Sends the email.

Args: msg (email.MIMEMultipart): The email to be sent.

rechnung.helpers.**str2bool**(*var*)

rechnung.invoice module

exception rechnung.invoice.**NoUnbilledItemsFound**
Bases: Exception

rechnung.invoice.**bill_cid_items**(*settings*, *contract*, *year*, *month*)
Creates billed items for the given month and year.

rechnung.invoice.**bill_items**(*settings*, *year*, *month*, *cid_only=None*, *dry=False*)
Bill all products for all customers (or just one) i.e. mark them to be included in the next invoice to be created.

rechnung.invoice.**create_billed_invoices**(*settings*, *suffix*, *cid_only=None*, *force=False*)
Bulk creates invoice yaml files for the customer from the customers billed items.

rechnung.invoice.**create_invoices**(*settings*, *year*, *month*, *cid_only=None*, *force=False*)
Bulk creates invoice yaml files for a specific month-year-combination.

rechnung.invoice.**fill_invoice_items**(*settings*, *items*)
Calculates the items which will appear on the invoice, as well as the total_gross, total_net and total_vat value.

rechnung.invoice.**generate_billed_invoice**(*settings*, *contract*, *suffix*)
Creates an invoice from the already billed items, i.e. collects all unbilled items adds it to the invoice, and enters the invoice number into the billed items.

It returns the invoice dict.

rechnung.invoice.**generate_invoice**(*settings*, *contract*, *year*, *month*)
Creates an invoice, i.e. calls the fill_invoice_items function, to get all the numbers right, as well as filling all the remaining required meta information about the customer.

It returns the invoice dict.

rechnung.invoice.**get_billed_items**(*settings*, *cid*)
Returns the billed_items for the given cid.

If there are no billed items yet, an empty list is returned.

rechnung.invoice.**get_period_from_item_keys**(*item_keys*)

rechnung.invoice.**iterate_invoices**(*settings*)
Generator which iterates over all contract directories and included invoice yamls, yields contract_invoice_dir and filename.

rechnung.invoice.**render_invoices**(*settings*)
Renders all invoices and saves pdfs to settings.invoices_dir.

rechnung.invoice.**save_billed_items_yaml**(*settings*, *billed_items*, *cid*)
Saves the billed items to the billed items file of the customer

rechnung.invoice.**save_invoice_yaml**(*settings*, *invoice_data*, *force=False*)
Saves the invoice_data to a yaml file in settings.invoices_dir.

rechnung.invoice.**send_invoices**(*settings*, *year*, *month*, *cid_only*, *force*, *suffix=None*)
Sends emails with the invoices as attachment.

For backwards compatibility: year and month are ignored, if suffix is given!

rechnung.settings module

exception rechnung.settings.**RequiredSettingMissingError**
Bases: Exception

If a setting is missing, but required to be set in the settings.yaml, this exception is thrown.

class rechnung.settings.**Settings**(*company_name*, *company_address*, *company_bank*, *contract_mail_subject*, *insecure*, *locale*, *password*, *sender*, *server*, *username*, *vat*, *invoice_mail_subject*, *assets_dir*, *contract_css_asset_file*, *contract_mail_template_file*, *contract_template_file*, *contracts_dir*, *csv_dir*, *delivery_date_format*, *invoice_css_asset_file*, *invoice_mail_template_file*, *invoice_template_file*, *invoices_dir*, *logo_asset_file*, *policy_attachment_asset_file*, *billed_items_dir*, *arrow_locale*)
Bases: tuple

arrow_locale
Alias for field number 26

assets_dir
Alias for field number 12

billed_items_dir
Alias for field number 25

company_address
Alias for field number 1

company_bank
Alias for field number 2

company_name
Alias for field number 0

contract_css_asset_file
Alias for field number 13

contract_mail_subject
Alias for field number 3

contract_mail_template_file
Alias for field number 14

contract_template_file
Alias for field number 15

contracts_dir
Alias for field number 16

csv_dir
Alias for field number 17

delivery_date_format
Alias for field number 18

insecure
Alias for field number 4

invoice_css_asset_file
Alias for field number 19

invoice_mail_subject
Alias for field number 11

invoice_mail_template_file
Alias for field number 20

invoice_template_file
Alias for field number 21

invoices_dir
Alias for field number 22

locale
Alias for field number 5

logo_asset_file
Alias for field number 23

password
Alias for field number 6

policy_attachment_asset_file
Alias for field number 24

sender
Alias for field number 7

server
Alias for field number 8

username
Alias for field number 9

vat
Alias for field number 10

exception rechnung.settings.**UnknownSettingError**
Bases: Exception

If a setting is found in the settings.yaml file which is unknown to rechnung, this exception is thrown.

rechnung.settings.**copy_assets**(*target_dir*, *orig_dir=PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/rechnu*
packages/rechnung-0.1-py3.7.egg/rechnung/orig'))
Copy the original assets, which are shipped with the tool, from the original directory (where the tool is installed) to the cwd (where the data is stored).

rechnung.settings.**create_required_settings_file**(*cwd*, *settings_file='settings.yaml'*)
Creates a settings file with all required settings listed, to be filled by the user accordingly.

rechnung.settings.**get_settings_from_cwd**(*cwd*, *create_non_existing_dirs=False*, *settings_file='settings.yaml'*)
Wrapper for get_settings_from_file to allow for easier exchange later.

rechnung.settings.**get_settings_from_file**(*settings_path*, *error_on_unknown=True*, *prepend_base_path=True*, *create_non_existing_dirs=False*)
Opens a settings.yaml and returns its contents as a namedtuple "Settings". It checks if all required settings are found in the settings file, as well if there are any unknown settings given. Finally the base_path is prepended to all settings ending with "_file" or "_dir".

rechnung.transactions module

rechnung.transactions.**parser_gls**(*csv_path*)
Parses CSV files from the GLS bank

rechnung.transactions.**read_csv_files**(*settings*, *year*, *month*)
Parses CSV files of a specific year/month combo

Module contents

- genindex
- modindex
- search

CHAPTER 2

Authors, Copyright, License

rechnung is developed by Florian Rämisch and Paul Spooren on behalf of *Westnetz w.V.* and *Reudnetz w.V.* two independent not-for-profit internet service providers (ISPs). It is licensed under the GNU General Public License v3. Copyright is by *Florian Rämisch and Paul Spooren, 2019*.

Python Module Index

r

Index