Work Pay Calculator

Luca Parolari[*]

2018/10/04

Abstract

WPC is a light-weight, highly configurable and easy to use library with a minimal CLI, with the objective of manage your work (hours and costs) and emit invoices.
This document contains a brief, requirements and design project for the library.

[*] luca.parolari23@gmail.com

Contents

1 WPC

1.1 What is

Work Pay Calculator has the objective to simplify the **management** of the work done in terms of hour, cost and profits. In other hands WPC is an application that smartly stores hours done for a work and allows to automatize the process of hours pay calculation and *occasional performance*[1] emission.

1.2 What is not

WPC is not a management tool for fiscal things, futhermore, for now, it's not an hour marker in terms of real-time start and stop counter (see 2.3).

1.3 Features

(a) Light-weight, simple and easy to use.

(b) Easily syncronizable with cloud services: only the executable and the data storage is needed. Data storage needs to be syncronized only if the SQLite engine is chosed.

(c) CLI inteface, in order to focus the objective to the application functionalities.

(d) Highly configurable: personalize environment variables and achieve your needs.

(e) Default answer configuration. Lot of configurable default answers, avoing boring data typing.

1.4 Examples

Scenario 1: today I've worked for a customer 6 hours: 3 of coding, 1 learing a new technology and 2 to publish the work online. What the hell, I have to register this things before I forget them.

No problem, open WPC and insert a new work in a way like this:

```
wpc> work: today 14:30 17:30 true
wpc> work: km? [0]:
wpc> work: add? [0]:
```

[1] This is an italian fiscal document that certifies work for someone.

```
wpc> work: registry? [General work]: I've coded the
pinco pallo project
wpc> Work inserted successfully!
```

And you can repeate this procedure any time you want to increase the datail of hours done in this day.

2 Analisys

This chapter will report the result of the project analisys, and its requirements.

2.1 Requirements

The system will be developed as a library written in Python (for educational purposes) with a minimal CLI[2]. The system will need a dababase, in this context SQLite will be chosen (for educational purposes again), however a strategy to make db engine interchangable could be adopted in later versions (see 2.3).

The system should allow the user to:

(a) Setup system variables, even if default values are setted. (see *2.2*);

(b) Manage clients;

(c) Enter a new line of work with following data:

   1) Date of work;
   2) Start and end time of work, or directly number of hours done;
   3) Boolean production value: true if the work can be marked as production, false otherwise;
   4) The kilometers done to reach the work place (if any);
   5) An add value, if any extra outgoings;
   6) Some notes, to mark down any noticeable data or to specify what the add value stands for;
   7) A description of work done;

(d) Mark some dates as paid;

[2] Command Line Interface

(e) Allow registering forfait payments, that should be removed from the next bill;

(f) Show some statistics:

1) Total hours done;
2) Total non-production hours done;
3) Total production hours done;
4) Total kilometers done;
5) Total add;

This should be allowed for each paid ”session”, including not paid data as default.

(g) Financial report:

1) Total hours profit;
2) Total non-taxable hours profit;
3) Total taxable hours profit;
4) Total kilometers cost;
5) Total add cost;
6) Total profit;
7) Total profit with fiscal elaboration (IVA, gross and net to pay).

Also this should be available for each paid ”session”, including not paid data as default.

2.2 Configuration

The system needs some static parameters that can change in the future. The a configuration manager is needed to handle this requirement.

Configuration to handle are:

(a) euro/hour value, i.e. how much an hour of work cost for the customer. For now, every hour has the same cost, but in future could be implemented a way to make some hours with a price and some other with other price;

(b) kilometers/litre, in order to calculate how miny litres of gas are needed base on km done;

(c) litre/euro, in order to calculate a possible value to expose as a cost (will be approximate);

(d) IVA, the italian fistal number for taxes.

2.3 Enhancements

Here will be listed all programmate or desiderable changes to the project.

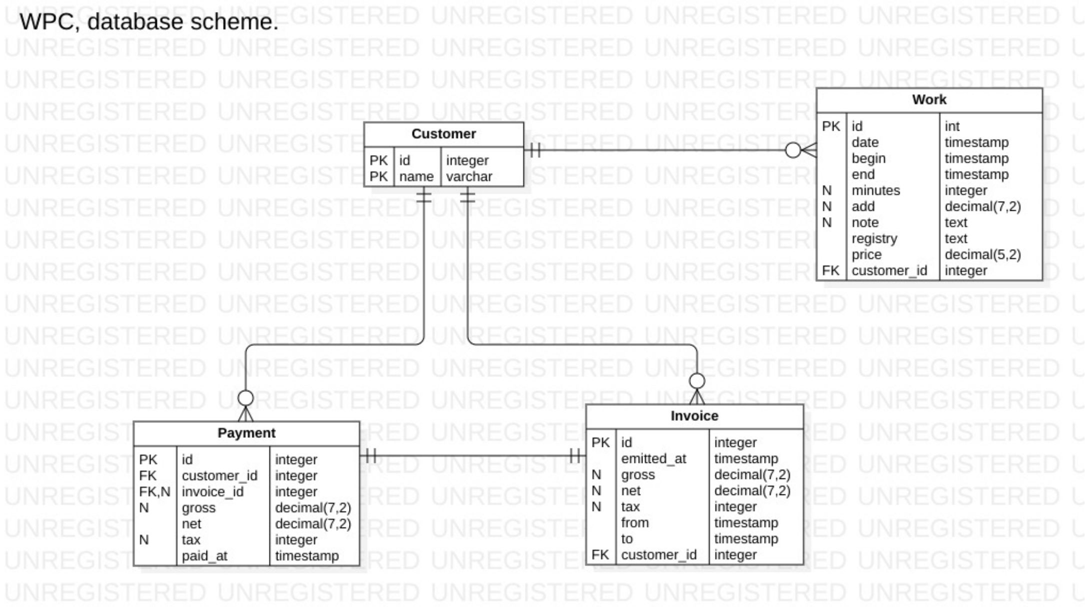
Programmed enhancements: *none* for now. To consider:

(a) Multiple database engine support;

(b) Different hour cost based on work;

3 Design

This section will describe how the software is designed.



WPC, database scheme.

Customer		
PK	id	integer
PK	name	varchar

Work		
PK	id	int
	date	timestamp
	begin	timestamp
	end	timestamp
N	minutes	integer
N	add	decimal(7,2)
N	note	text
	registry	text
	price	decimal(5,2)
FK	customer_id	integer

Payment		
PK	id	integer
FK	customer_id	integer
FK,N	invoice_id	integer
N	gross	decimal(7,2)
	net	decimal(7,2)
N	tax	integer
	paid_at	timestamp

Invoice		
PK	id	integer
	emitted_at	timestamp
N	gross	decimal(7,2)
N	net	decimal(7,2)
N	tax	integer
	from	timestamp
	to	timestamp
FK	customer_id	integer

3.1 CLI

The command line interface is a foundamental tool: it allows the usage of the program without any complicance in UI design, however, even the CLI needs to be designed.

Commands for clients management:

```
> cli show
> cli add <id>|<name>
> cli remove <id>|<name>
> cli switch <id>
```

Commands for work management:

```
> (cli) work --filters column1=value1, column2=value2, ..., columnN=valueN
> (cli) work add
> (cli) work remove <id>|<date>
> (cli) work edit <id>|<date>
```

Commands for invoices management:

```
> (cli) inv --filters column1=value1, column2=value2, ..., columnN=valueN s
> (cli) inv add
> (cli) inv remove <id>
> (cli) inv edit <id>
```

Commands for payments management:

```
> (cli) pay --filters column1=value1, column2=value2, ..., columnN=valueN s
> (cli) pay add
> (cli) pay remove <id>
> (cli) pay edit <id>
```

Every command will start guided procedure throught actions.

4 Version

4.1 Analisys

Analisys version **0.4.0**. Requirements are a draft, might change. No new official version will be issued until the end of the draft.

4.2 Design

Design version **0.2.0**.

5 Authors

Luca Parolari (luca.parolari23@gmail.com), computer science' student at Unversity of Parma, Italy.

5.1 Collaborating

Contact me at luca.parolari23@gmail.com, or contribute directly on GitHub.
If you find an issue please report it on GitHub.

6 License

GNU/GPL v3, or any later versions.