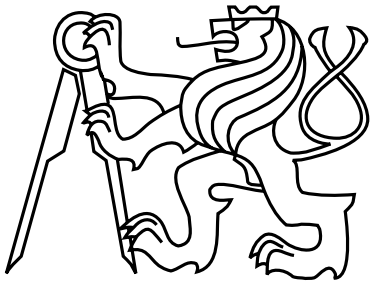
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Bachelor's Project

Developing an accounting component for a car-sharing support information system

Jakub Ječmínek

Supervisor: Ing. Martin Komárek

Study Programme: Software Engineering and Management

Field of Study: Software Engineering

January 2, 2013

Aknowledgements

I would like to thank my parents and grandparents for support, then I'd like to thank my supervisor Ing. Marin Komárek for accepting my request to join Metrocar team. Also I have to mention Bc. Petr Pokorný who gave me a lot of advices and recommendations about Django/Python development.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Zábřeh na Moravě on December 21, 2012 .....

Abstract

The goal of this work is to continue in development of web application Metrocar, which will be used for operating carsharing company. It addresses analysis, design and implementation of application which connects system Metrocar and accounting system Flexibee. Information system Metrocar is developed on Python platform and framework Django.

As a part of implementation of application which connects system Metrocar and system Flexibee is developed library Flexipy in Python. This library is using REST API which is a part of system Flexibee.

Abstrakt

Cílem práce je pokračovat ve vývoji webové aplikace Metrocar, která bude sloužit k provozování carsharingové společnosti. Zabývá se analýzou, návrhem a implementací aplikace, která propojuje systém Metrocar s účetním systémem Flexibee. Informační systém Metrocar je vyvíjen na platformě Python za použití frameworku Django.

V rámci implementace aplikace pro komunikaci mezi systémem Metrocar a systémem Felxibee je vyvíjena i knihovna Flexipy pro komunikaci se systémem Flexibee v jazyce Python. Tato knihovna využívá REST API, které je součástí systému Flexibee.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Carsharing definition

Carsharing is model of renting cars where group of people is sharing car for a short periods of time. This is especially usefull for people who use car only occasinally. The organization that provides carsharing service don't have to be necessarily comercial company, it can be cooperative, public agency or ad hoc grouping[12].

Carsharing model of renting cars is today common in North and Western Europe. In Czech Republic there is one company that provides this kind of services for city Brno[11]. Therefore there is still markateplace for another company that would provide carsharing services.

1.2 History of the project

Project Metrocar is under development sicne year 2008. Original implementation of the project was part of a bachelor thesis by Bc. Ondřej Nebeský. His solution was implemented in PHP and CMS Drupal. This implementation allowed users to view cars on the map, create reservation for car and monitor their journeys. It also provided mechanisms for creating and managing carsharing subsidiaries[6].

The final result didn't met all the requirements and also chosen architecture wasn't optimal for the planned deployment of the system. These shortcomings where addressed in the master's thesis of Ing. Filip Vařecha. His goal was to improve and transform this system. The main goal was to create public API fot the system. As the result, the whole system was rewritten from scratch. Python and Django web framework were used as primary technoliges for the resulting system. The complete list of changes and new features cna be found in the thesis of Filip Vařecha[10].

Next development on the project was done as part of bachelor thesis by Bc. Jan Wágner. His main goal was to finish system for reservations and user managment. He was also responsible for selecting hosting for the application. Further details about his work on the project can be found in the resulting thesis[13]. There were also some small contributions to the project by students of the subject Y36SI2. The complete list of their work and changes can be found website of the project in the final reports[3].

Last contributor to the project was Bc. Petr Pokorný. He was primarly focused on the comunication between the server side of the application and car units. He was also responsible for extreme improvment in documentation of the project and updating project's dependecies. Thansk to his vast knowledge of Python and Django he has become importent figure of the project and excelent advisor.

Chapter 2

Problem description, goal specification

2.1 Problem description

When I joined the Metrocar project, there was only partial solution for accounting in the system. The solution was capable only to create invoices and invoice items. For printing to PDF files and sending invoices to users, there was small Python script that was supposed be running periodicly by cron and check if some invoices are supposed to be sent to the users. Also there was no way for users to gain creidt, which is basicaly curency in our system. There was alo no solution for checking state of company's account. Therefore the state(active, overdue and paid) of every invoice would have to be changed manually by manager of the company. This would force manager to spend to much time doing just checking statuses of invoices. Also this aprocha would probably fail for a larger group of users. Therefore it was decided, that system for accounting in the project has to be improved.

2.2 Goal specification

Design and implement application that provides connection of system Metrocar[1] to accounting software Flexibee[2]. Systems Metrocar and Flexibee will use this application for communication and exchange of data. This application will use REST API, which is provided by Flexibee[3] Official goal specification is as follows:

- Identify requirements for the application.
- Create UML models of proposed application.
- Implement the application.
- Create unit tests, functional tests and performance tests).
- Create documentation for the application.

Before we decided to use for our project Flexibee as an accounting system, I was doing as a par of the subject A7B16PRO a research about possibilities how to connect system Metrocar to hombanking solution and accounting system. As a result of this research, it was decided that we will use Flexibee. I decided to attach the output of my research about hombanking systems and accounting systems on Czech market to this report, because it served as a foundation for a decision to choose Flexibee.

2.2.1 Online banking

Originally was system Metrocar without any kind of connection to company's bank account. Therefore, manager of company would have to manually check incoming payments and then change the status of appropriate invoice. This is very bad solution and it would be very time consuming for manager. The team that was working on the project in the year 2010 started to work on the solution, that would require create python script that would manually parse the informations about payment from email or sms that woudl be sent from bank[5]. As a part of my research I found only this article about there ideas for this solution, but I wasn't capable to found any source code. Therefore I suppose they didn't impement this idea.

At first I thought that this solution would be great for system Metrocar. I also found out that owner and founder of the Roští.cz[1] Adam Štrauch created Python project that provides similar solution. His module is capable to log in to online banking system of mBank company and from there download informations about incoming payments in CSV format and then parse them into Python data structures. His project is released under BSD license and is publicly available on github[2]. Article about the development of this module and the reasons for its creation can be found on his personal website[1]. This solution is not optimal for out project, because as even author has stated, if mBank change some part of their online banking system, the script will simply stop to work. This is also not exactly recomended aproach to obtain this kind of informations. Typically every bank istituion have their own system, which allowes clients to download certain informations about payments, account status etc. These systems are called hombanking systems. There are several standard formats which these systems usually use. The complete list can be found here[**?**]. From my experience in these days the most used ones are Gemini, MultiCash and ABO. The basic functionality of these systems is usually same, therefore I will further describe only what Gemini in version 5.0 provides, other format support this too. Banking operations:

- Domestic payments and direct debit orders.
- International payments.
- Domestic and international payments with future value date.
- Urgent payments.
- Permanent payment orders with automatic creation of a pre-set payment calendar.
- Time deposits.

[1] company that is providing hosting for project Metrocar(autonapul.cz)

Security features:

- Password for electronic signature.
- Individual user passwords and access rights.
- Authorization before sending payment orders to the bank.
- Private and public RSA key.
- Electronic signatures are automaticly checked by bank institution.

Gemini has other usefull features but not all of them are supported by every hombanking system. The most resourcefull documentation about Gemini has Raiffeisen Bank[9] from which I learned about this format. Therefore these other special features might be implemented only in Raiffeisen Bank homebanking system.

Other features of Gemini:

- Export/import of payment orders to/from accounting system.
- Template of payment orders.
- Automatic communication with bank.
- Current exchange rates Raiffeisenbank Inc. and ČNB.

There are also other featuers, for complete list visit[9]. When I joined the Metrocar team as a part of the A7B16PRO subject, it was not yet decided which bank company will be used for Metrocar company's bank account. Therefore it was decided that my next assigment will be to select suitable accounting system for system Metrocar, which will already support communication and infromation exchange with homebanking system and that this communication will be using Gemini format and other formats too if possible.

2.2.2 Accounting system

I was responsible for selecting appropriate accounting system for our project by the superviser Ing. Martin Komárek. I created a list of requiremnets which we expected from the accounting system.

- Capability to comunicate and exchnge data with homebanking system.
- The accounting system should have interface(for access from other systems).
- The accounting system should have sufficient documentation for developers of information systems.
- The aforementioned documentation should be free of charge.

After I had this specification about what kinf of we were looking for, I started to search for the system on Czech market. A lot of people were telling about system Pohoda[7]. After small research I found that system Pohoda allows imports and exports of data through exchange of XML files[8]. XSD files and example XML files are publicly avaliable and free. There is also short documentation for developers, but I found it too simple. The biggest problem with system Pohoda was tha fact that for testing of my component, I would have to purchase a license for the system. Also a lot of people who are working in companies that are using system Pohoda told me that they have negative experience with the system.

So I started to look for other systems, until I spoke with teammate Petr Pokrný, who recommended me accounting system Flexibee[4]. After some research I found out that Flexibee is the only accounting system on Czech market that offers completely open REST API for developers with extensive documentation, examples and tutorials. They also provide publicly available cloud version of Flexibee which developers can use for testing of their applications. Other possibility of testing is to register on their website as a developer and then they will provide full version of their system for free. They also have mailing list which they use to announce new releases and changes in their system. This is perfect ecosystem for developers. The REST API supports both XML import/export or JSON. Strongest side of Flexibee is documentation for developers.

After I familiarized myself with Flexibee adequately I presented it to the supervisor as a solution that I would recommend and that seemed to me as the most suitable to our project and our needs. Also after discussion with Petr Pokorný who already had experience with connecting information system to Flexibee, we suggested to use Flexibee in cloud insteadof deployment on our server. This was approved by our supervisor Ing. Martin Komárek and I began to analyse existing application[2]Invoices and what would be the best approach for connection of accounting system Flexibee.

[2]Application in terms of Django application https://docs.djangoproject.com/en/dev/intro/tutorial01/#creating-models

Chapter 3

Analysis and Solution Proposal

In this chapter I will provide analysis of my solution on how to connect existing system Metrocar to an accounting system Flexibee. I will describe reasons for chosen approach and technologies.

3.1 Invoices application

Application invoices in project Metrocar has responsibilities for CRUD operations with invoices and other accounting documents. It also contains tools for printing invoices and sending them to users(customers). This application was originally created by Filip Vařecha and then maintained by Jan Wágner. As I mentioned before, this solution has many shortcomings. Therefore as a part of this project I am supposed to connect system Metrocar to system Flexibee. At first I planned to remove application Invoices and replaced it by new application that would be directly connected to Flexibee. However as it turned out this application is very integrated into the Metrocar system. Also the standard things like managing invoices is well hnadled by the application. Flexibee is very good system and has a lot of happpy customers, but even systems like this expoerience from time to time poblems. Petr Pokorný expereinced a few server crashes while he was working on the project that involved Flexibee as an accounting system. At that point the whole system is inaccessible for users. because they can't create invoices or issue their bills. A this point keeping and old Invoices application could help. When a user create invoice, or issue a bill or just do some action that involves accounting and at this point Flexibee crashes, accounting module(later described) will notice this and set certain actions into motion. Accounting module will use old Invoices application and store the canges or new data in local database in apropriate tables. Also he notifes manager that server on which is Flexibee deployed has crashed. After server with Flexibee is running again, accounting module will resolve incosistences between local database and Flexibee's recors. This way we can say that old Invoices application will serve as some kind of fail-safe mechanism, which will be primarly used to serve users' requests while Flexibee is down. For this reasons I've decided to keep old application Invoices in the project Metrocar. It will be slightly modified and certian functionalities will be transfered on to Flexibee(for example printing of invoices and sending them to users).

3.2 Accounting module

As was metioned above, the accounting module will be responsible for handling situations in which Flexibee has crashed, but this will be only secondary purpose of this module. The main purpose of this module will be to act as intermidiate between Metrocar system and Flexibee. It will define interface for Metrocar system through which will system Metrocar interact with Flexibee. Thanks the nature of Python, this module can be easily replaced in the future with diffrent implementation of the same interface, so on the side of Metrocar system, it will look like that nothing has changed, but actually Metrocar system will now comunicate with completely diffrent accounting system on the other side. Therefore there will be no coupling between Metrocar system and Flexibee. This is very important because if the interface of accounting module was coupled with Flexibee it would extremely hard and tedious to change accounting system.

On the side of Flexibee system, the accounting module will use library flexipy which will be described in the next section. This will also help to adhere separation of concerns pricniple[**?**], because accounting module will simply call function from flexipy and passes data if needed. It will not know anything about Flexibee REST API or format of data in which Flexibee comunicate.

3.3 Flexipy

Chapter 4

Implementation

Popis implementace/realizace se zaměřením na nestandardní části řešení.

Chapter 5

Testing

- Způsob, průběh a výsledky testování.
- Srovnání s existujícími řešeními, pokud jsou známy.

Chapter 6

Summary

- Zhodnocení splnění cílů DP/BP a vlastního přínosu práce (při formulaci je třeba vzít v potaz zadání práce).
- Diskuse dalšího možného pokračování práce.

Bibliography

[1] Adam Štrauch. *Jak jsem si dělal API pro internetové bankovnictví mBank* [online]. 2012. [cit. 23. 12. 2012]. Dostupné z: <http://initd.cz/posts/jak-jsem-si-delal-api-pro-internetove-bankovnictvi-mbank/>.

[2] Adam Štrauch. *python-mbank - mBank Python parser* [online]. 2012. [cit. 23. 12. 2012]. Dostupné z: <https://github.com/creckx/python-mbank>.

[3] Assembla. `https://www.assembla.com/spaces/wagnejan_metrocar/wiki`.

[4] Flexibee. Flexibee accounting system. `http://www.flexibee.eu/`.

[5] Metrocar team 2010. *eBanking a SMS* [online]. 2010. [cit. 26. 12. 2012]. Dostupné z: <https://www.assembla.com/spaces/metrocar/wiki/eBanking_a_SMS>.

[6] NEBESKý, O. *Bakalářská práce: Rezervační systém carsharingové společnosti.* KP FEL ČVUT, 2009.

[7] Pohoda. Accounting system Pohoda, . `http://www.ucto-pohoda.cz/`.

[8] Pohoda. XML communication with accounting system Pohoda, . `http://www.stormware.cz/xml/`.

[9] Raiffeisen Bank. *Gemini 5.0* [online]. 2012. [cit. 23. 12. 2012]. Dostupné z: <http://www.rb.cz/firemni-finance/velke-podniky/platebni-styk-a-cash-management/elektronicke-bankovnictvi/gemini-5-0/>.

[10] VAřECHA, F. *Diplomová práce: Transformace informačního systému pro carsharing do webové služby.* KP FEL ČVUT, 2010.

[11] web:brno. Autonapůl Brno. `http://www.autonapul.org/`, stav z 30. 12. 2012.

[12] Wikipedia contributors. *Carsharing* [online]. 2012. [cit. 23. 12. 2012]. Dostupné z: <http://en.wikipedia.org/wiki/Carsharing>.

[13] WáGNER, J. *Bakalářská práce: Příprava systému autonapul.cz na reálné nasazení.* KP FEL ČVUT, 2012.

Appendix A

Seznam použitých zkratek

XML Extensible Markup Language

JSON JavaScript Object Notation

API Application programming interface

REST Representational state transfer

CSV Comma-separated values

⋮

Appendix B

UML diagramy

Tato příloha není povinná a zřejmě se neobjeví v každé práci. Máte-li ale větší množství podobných diagramů popisujících systém, není nutné všechny umísťovat do hlavního textu, zvláště pokud by to snižovalo jeho čitelnost.

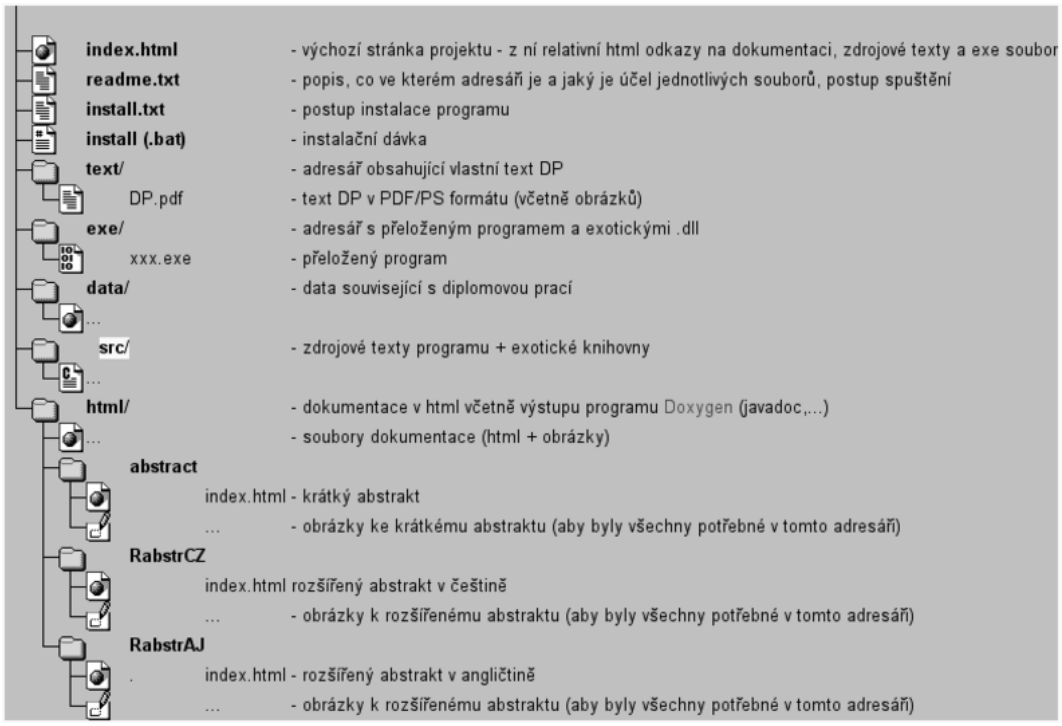


Figure B.1: Seznam přiloženého CD — příklad