Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

wgfinance.ch

Group Thesis

Dominik Fankhauser
Jonas Luder
Lorenz Koestler
Raphael Seebacher

{forename}@wgfinance.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Barbara Keller
Prof. Dr. Roger Wattenhofer

October 19, 2011

Abstract

This group thesis presents *wgfinance.ch*, the *versatile and intelligent finance tool for you and your flatmates.* From the history of wgfinance to the most current version, the concepts behind wgfinance are explained and implementational details are presented. Furthermore design and usability issues are looked at, as well as potential competitors. Finally a roadmap for the future is provided as a development guideline.

Contents

Chapter 1

Introduction

In this introductory chapter the point we started from is being illustrated. The first view on the new wgfinance is given in chapter 2, where the basic concepts and features we planned are discussed. A more concrete point of view is provided in chapter 3, which focuses on the implementation of the previously specified features, as well as on design and usability. Security issues one faces when developing a publicly accessible website are treated in chapter 4. Rivals and competitors are looked at and compared to wgfinance in chapter 5. We conclude this thesis in chapter 6, where we provide a plan for future development of wgfinance.

1.1 A Short History of wgfinance

It all started in midsummer 2009, when Lorenz and some of his friends moved to a flat. As in every flat, the finances are a very important issue. Being a part-time programmer for an ETH spinoff company, Lorenz quickly realized that doing all the accounting by hand or using a spreadsheet was far too impractical and kind of old-fashioned. There was a need for something new. He therefore spent some night shifts developing a quite simple, but never the less useful webbased tool: The very first version of wgfinance was born. One might of course ask, why Lorenz didn't look out for some preexisting tool, but by writing the tool from scratch one can precisely tailor the tool to ones needs, which is basically why Lorenz wrote wgfinance himself.

Soon after Lorenz has developed wgfinance and registered the domain wgfinance.ch, Jonas' flat began doing their accounting using wgfinance. In late 2009 Dominik, Jonas, Lorenz and Raphael first met to discuss how to continue with wgfinance. First concepts and ideas were brought to paper, but it took another year for the developing of the new wgfinance to begin:
In late fall 2010 we found ourselves looking for a group thesis for the upcoming spring term. Some day one of us came up with the idea of developing wgfinance as our group thesis. It didn't take long until we were all convinced to do so,

since we always wanted to develop wgfinance further but ended up having not much spare time.

1.2 The Old wgfinance

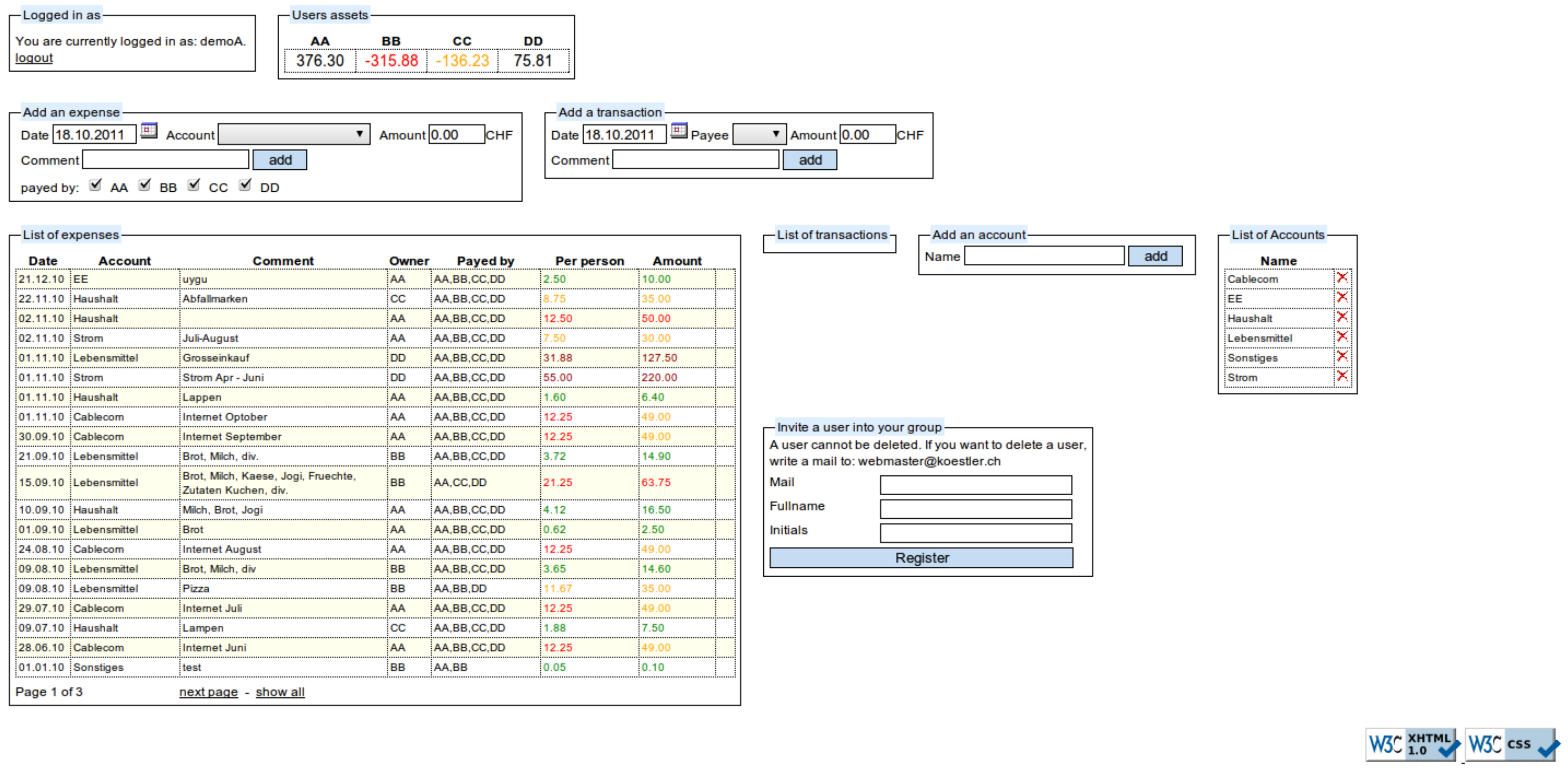
In order to give an answer for the legitimate question of why developing a totally new wgfinance, we have to look at the architecture and the concepts behind the very first version of wgfinance. Needless to say, as this first version of wgfinance was a product of several night shifts, its capabilities were limited and yet it provided the very basic functionality for accounting.

It was basically a one page website consisting of various boxes (cf. figure 1.1). For each basic task, such as adding an expense, or showing a list of expenses, there was a specific box. All one could do with the old wgfinance was collecting and viewing expenses and transactions, setting up accounts, and calculating the overall credit each member of the flat had.

From a technical point of view the old wgfinance based on PHP and MySQL and was written using no specific framework, but rather some proprietary source code from the ETH spinoff company Lorenz works at.

1.3 The Need for a New wgfinance

The main reason why we started completely from scratch with the new wgfinance, is because it partly consisted of proprietary source code, as already mentioned above. It would not have been possible to launch wgfinance as a public product when it still consisted of that proprietary code. A further, yet minor reason was that the first wgfinance didn't make use of a framework and therefore many basic functionalities had to be implemented by hand. Furthermore the old wgfinance lacked some useful functionalities regarding user management, account management, statistics and more. The need for developing a new version of wgfinance is, hence, justified for the aforementioned reasons.



Logged in as

You are currently logged in as: demoA.
logout

Users assets

AA	BB	CC	DD
376.30	-315.88	-136.23	75.81

Add an expense

Date 18.10.2011 Account Amount 0.00 CHF
Comment add
payed by: ✓ AA ✓ BB ✓ CC ✓ DD

Add a transaction

Date 18.10.2011 Payee Amount 0.00 CHF
Comment add

List of expenses

Date	Account	Comment	Owner	Payed by	Per person	Amount
21.12.10	EE	uygu	AA	AA,BB,CC,DD	2.50	10.00
22.11.10	Haushalt	Abfallmarken	CC	AA,BB,CC,DD	8.75	35.00
02.11.10	Haushalt		AA	AA,BB,CC,DD	12.50	50.00
02.11.10	Strom	Juli-August	AA	AA,BB,CC,DD	7.50	30.00
01.11.10	Lebensmittel	Grosseinkauf	DD	AA,BB,CC,DD	31.88	127.50
01.11.10	Strom	Strom Apr - Juni	DD	AA,BB,CC,DD	55.00	220.00
01.11.10	Haushalt	Lappen	AA	AA,BB,CC,DD	1.60	6.40
01.11.10	Cablecom	Internet Optober	AA	AA,BB,CC,DD	12.25	49.00
30.09.10	Cablecom	Internet September	AA	AA,BB,CC,DD	12.25	49.00
21.09.10	Lebensmittel	Brot, Milch, div.	BB	AA,BB,CC,DD	3.72	14.90
15.09.10	Lebensmittel	Brot, Milch, Kaese, Jogi, Fruechte, Zutaten Kuchen, div.	BB	AA,CC,DD	21.25	63.75
10.09.10	Haushalt	Milch, Brot, Jogi	AA	AA,BB,CC,DD	4.12	16.50
01.09.10	Lebensmittel	Brot	AA	AA,BB,CC,DD	0.62	2.50
24.08.10	Cablecom	Internet August	AA	AA,BB,CC,DD	12.25	49.00
09.08.10	Lebensmittel	Brot, Milch, div	BB	AA,BB,CC,DD	3.65	14.60
09.08.10	Lebensmittel	Pizza	BB	AA,BB,DD	11.67	35.00
29.07.10	Cablecom	Internet Juli	AA	AA,BB,CC,DD	12.25	49.00
09.07.10	Haushalt	Lampen	CC	AA,BB,CC,DD	1.88	7.50
28.06.10	Cablecom	Internet Juni	AA	AA,BB,CC,DD	12.25	49.00
01.01.10	Sonstiges	test	BB	AA,BB	0.05	0.10

Page 1 of 3 next page - show all

List of transactions

Add an account

Name add

List of Accounts

Name	
Cablecom	✗
EE	✗
Haushalt	✗
Lebensmittel	✗
Sonstiges	✗
Strom	✗

Invite a user into your group

A user cannot be deleted. If you want to delete a user, write a mail to: webmaster@koestler.ch

Mail
Fullname
Initials
Register

Figure 1.1: The first version of wgfinance.

Chapter 2

Specification

In software development a quite well proven approach is to first fully - or at least mostly - define the concepts and features a piece of software is supposed to have, and then actually implement it. Below we illustrate the process just described, by first stating the assumptions we acted on, then specifing the features, the concepts and finally some technological details. The descriptions below are deliberately held in a rather abstract, non-implementation-specific manner, as the next section focuses on implementation.

We define a feature to be something the user actually sees and can make use of, whereas a concept is how wgfinance accomplishes certain tasks in the background, hence not directly visibile to the user.

2.1 Assumptions

As mentioned above, we decided on the assumptions stated below, in order to simplify our programming task on one hand, and to be able to provide tailor-made features on the other hand. Note that the following assumptions do not impose major constraints.

- We assume that the *flatmates trust each other*. In our opinion this trustfulness is necessary, since wgfinance is designed to keep track of expenses and not to settle disagreements between flatmates.
- As the members of flat shares can change quite frequently, we further assume that a *user can be a member of multiple flat shares* at a given time instant. In our opinion this is quite useful, when moving from one flat to the next, while still having unsettled expenses with the former flatmates.
- To mitigate potential currency conversion problems, especially regarding currency exchange rates over time, we decided that a *flat share has got only one currency*. From our point of view, the case where several currencies are needed on a daily basis is negligible.

2.2 Features

With the above assumptions, we can now specify the features we want to implement. We tried to limit these features to a minimun, such that wgfinance provides the basic functionality, and to implement selected features, if additional resources were available. We decided to implement the features listed below.

- **Expense Tool**
  The most obvious feature wgfinance should provide is an easy to use, but yet versatile tool for entering expenses. In order to guarantee the versatility of this expense tool it has not only to provide input fields for date, comment, amount, account, and checkboxes for those flatmates who pay, but rather expense splitting functionality to guarantee much faster entering of expenses. An expense has to be splitted when a receipt has multiple items, for which not every flatmate has to pay, and which belong to different accounts.

- **Statistics**
  The counterpart to the entering of expenses is the viewing of single and/or grouped expenses in various manners. We therefore planned on three different types of what we refer to as *statistics*: Pie charts, bar charts and tables. These statistics have to be configurable regarding the details of expenses, such as date, account, amount, etc., as well as regarding the time domain and the time interval.

- **Shopping List**
  Not absolutely necessary, but easy to implement and useful for users is the shopping list feature. The key idea of this feature is to give the users the possibility to add multiple lists, to which they can add items to be bought. These items can be marked as bought, and can be deleted. We also planned to give a shopping list a scope: The creator of a list can choose, whether the list can be seen by the whole flat share, or only by himself. This feature would become particularly useful in combination with smartphones, as lists could be modified in real-time, while being at e.g. a supermarket.

- **Management and Logging**
  Last but not least are several features regarding user and flat management, and logging such as

  - functionality to reset a forgotten password
  - user property changes
  - managing accounts
  - adding, modifying, deleting members of a flat share
  - having all relevant action logged

At this point we would like to stress that, even though functional with the above features, there are several features that have already been thought on. A list of possible future features is provided in section 6.

2.3 Concepts

2.3.1 Basic Concepts

First of all, there are some basic, not really project specific concepts many software projects implement and all of them are usually provided by a framework (cf. 2.4.2). These basic features consist of providing the *Model-View-Controller pattern*, *Object-Relational Mapping*, *Internationalization*, *Modularization*, *Extendability* and various other. It therefore makes sense to base the tool on a well-known and well-documented framework with a reasonably big community.

2.3.2 Accountancy

A key question we had to address is how to store all financial data in a proper, but flexible and easy to use way. Even though we could have implemented our very own scheme, we decided to implement a standard double entry bookkeeping. The issue of how to map such a bookkeeping onto a relational database has proven to be already solved[1].

2.3.3 The wgfinance API

To fully exploit the above specified shoppinglist feature, we planned on an *application programming interface* for external applications to provide access to the shoppinglist feature. For reasons of simplicity in parsing and readability, we decided to use HTTP GET parameters within an url[2] for queries by a client and the XML format for replies by the server.

2.4 Technological Specification

2.4.1 Python

We decided to use the python programming language [3] for the following reasons:

[1]Double Entry Accounting in a Relational Database `http://homepages.tcp.co.uk/~m-wigley/gc_wp_ded.html`

[2]cf. RFC 1738 `http://www.rfc-editor.org/rfc/rfc1738.txt`

[3]`http://www.python.org`

- It is a very widely used programming language, with a useful online documentation and a very large and active community.
- Python has its own object-oriented concept, which is very powerful. Furthermore, python relies on intendation rather than on brackets for structuring code. This leads to very readable and clean code, which, from our point of view, is an advantage.
- No one of the wgfinance team had yet gotten in contact with python, which is an advantage on one hand, and a disadvantage on the other: We all wanted to learn a new programming language, even though we would have been much faster in PHP, as some of us had already developed using PHP.

2.4.2 The Django Framework

Having decided to use the Python programming language, we had to find a suitable framework, as already discussed above. Quite quickly we took the decision to use the Django framework. According to their website[4]

> Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Django implements the *model-view-controller pattern*, to which they rather refer to as the *model-template-view pattern*. It has many users, many of which are active on the project's mailing list. In addition django is being constantly developed further and is very well documented. Built-in in django are various functions as for example a template engine, form classes, internationalisation, and more.

2.4.3 The Database

For the database in the background we chose MySQL[5], which is an open-source and well documented relational database. MySQL is a very powerful database, that provides more than enough functionalities and performance than we needed for wgfinance. Moreover we already used MySQL several times and, hence, already have some practical experience.

[4]`https://www.djangoproject.com`
[5]`http://mysql.com`

2.4.4 Libraries

- **JQuery**[6]
  JQuery is an open-source JavaScript library that allows us to manipulate HTML objects in a much simpler and more efficient way. Due to the fact that it's widely used, it is very stable and mostly free of bugs. Furthermore it is cross-browser compatible. JQuery in particular provides functions to select, manipulate and animate HTML objects, as well as to interact with the server using Ajax requests.

- **jqPlot**[7]
  jqPlot is an open-source JQuery plugin that is capable of displaying various types of charts, including line plots, bar charts and pie charts. The plugin provides various options for customization.

- **JQuery UI**[8]
  JQuery UI is an official extension to the JQuery framework, which provides the functionality to implement various types of user interface widgets, such as calendar widgets, dialog windows, progress bars, etc.

- **DataTables**[9]
  DataTables is another plugin for the JQuery library. It provides quite sophisticated enhancements to HTML tables, such as internationalization, multi-column sorting, search functions and many more.

- **JSTree**[10]
  Finally, JSTree implements a tree object, which is based on JavaScript and provides many features such as drag and drop support, inline editing, and so on.

[6]`www.jquery.com`
[7]`www.jqplot.com`
[8]`http://jqueryui.com`
[9]`http://datatables.net/`
[10]`http://www.jstree.com/`

Chapter 3

Implementation

3.1 How-To Develop

- extensive use of a mailinglist
- subversion
- wiki with a few howtos (setup of working station etc.)
- setup of working station: ubuntu, local testserver, sqlite and alternatively mysql

3.2 Backend

3.2.1 Models

3.2.2 Views

3.2.3 Client/Server Interaction

3.2.4 The Forms Class

3.3 Frontend

3.3.1 Templates

The Django template engine gives the programmer the possibility to dynamically compose the pages by building generic templates for various sub-pages and linking them together. Templates can include other templates, or also extend base templates. They can contain logic variables and loops which are then interpreted by the template engine. The placeholders in the templates are filled by python, before the server delivers the page to the client. This has several advantages: On one hand it is possible to build very complex pages while on same time keeping the templates simple and generic. This especially means, that design changes usually require changes in just a few files. ¡Der Aufbau der Seite kann sehr einfach kontrolliert werden¿

Each page extends a base template, and is composed of a header, a content, and a footer. On the main page the content is composed of several boxes which contain a certain content (for example a shopping list, or the statistics) and can include tabs. The contents and sizes (wide or narrow) of the boxes can be defined in python. The same principle also applies to the settings page, where there are tabs for each main category (personal settings and settings that apply to the whole flat share). Each tab contains various boxes for each subcategory.

3.3.2 Functionalities

Using JQuery

As mentioned above, JQuery is a widely used JavaScript library. HTML objects are selected using CSS[1]-like selectors, which means that it is much simpler to select for example all elements of one class than it is using only JavaScript. Once the objects are selected, various functions can be applied on them, for event handling purposes or to manipulate their attributes, amongst others. We used JQuery extensively for most user interface functionalities of the page, for example:

- implementing the tabs on the main page, as well on the settings page
- in the expense box
- in the shopping list box

Statistics using jqPlot

To draw the bar and pie charts we used the jqPlot library. The data to be displayed is delivered to the client in JSON[2] format and then parsed by JavaScript. This data was then used to compute various parameters used to customize the appearance of the plot. We for example calculated the ticks on the y-axis ourselves (instead of using the standard autoscale function provided by jqPlot) since we wanted to display 'nicer' numbers, i.e. multiples of powers of ten. The data was then handed over to the jqPlot constructor which rendered the plot and displayed it. The statistics are automatically refreshed if the user enters an expense, or if he switches tabs.

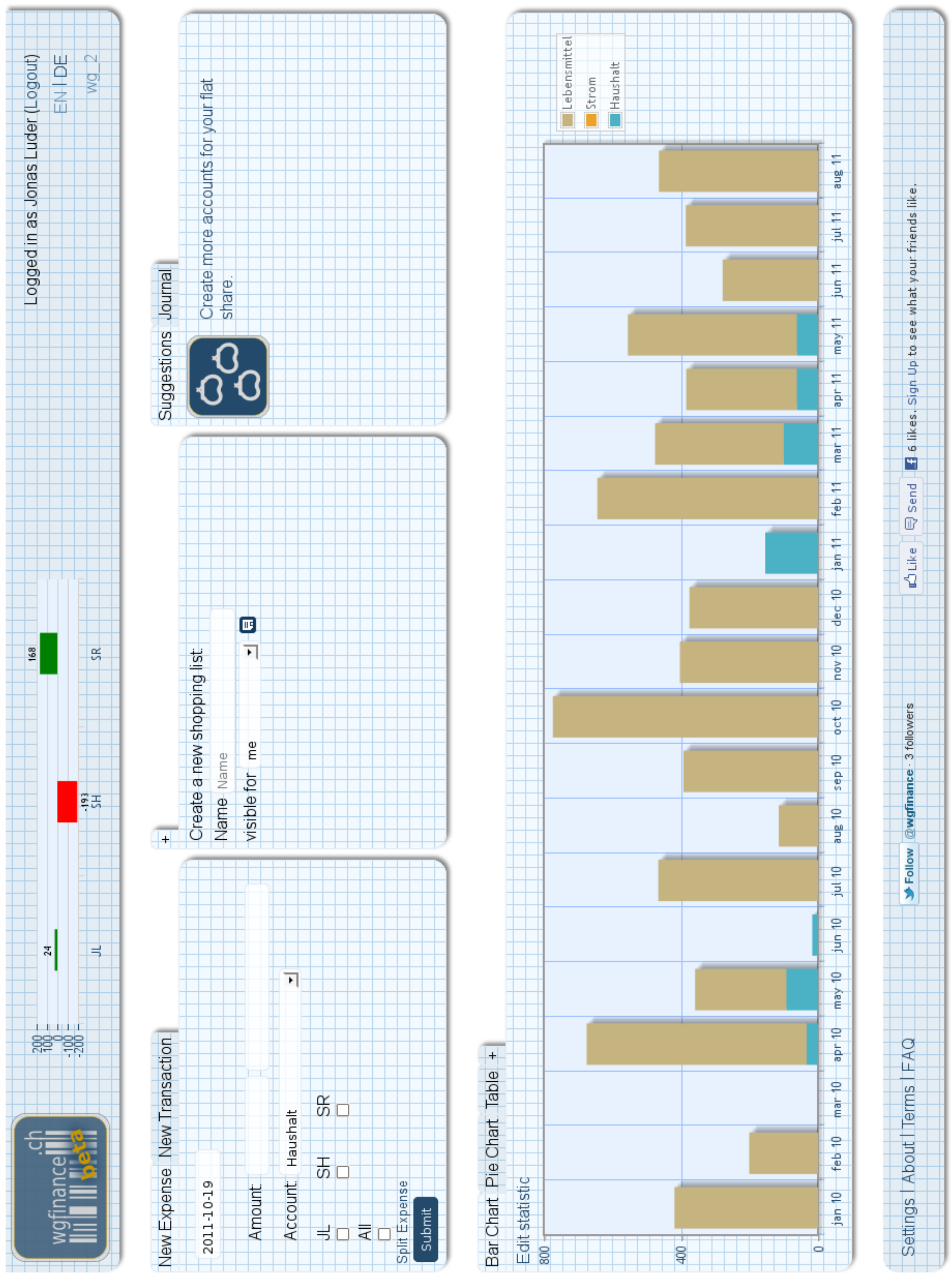
[1] `http://www.w3schools.com/css/default.asp`
[2] `http://www.json.org/`

3.4 API

Regarding the use of the wgfinance application programming interface, we refer to the appendix, where a detailed usage scheme for the API is given.A.1

3.5 Design and Usability

In this section we'd like to describe the basic ideas behind the design. Our goal was to have a tool, that can be used with as few clicks as possible, and that consists of as few pages as possible. However, the tool has still to be usable and clearly laid out.
To achieve this goal, we came up with the boxes design. For every tool there exists a box. The boxes can be displayed next to each other on the main page. Furthermore, we changed the content dynamically using AJAX. Therefore no page reloads are necessary, which makes the page slightly faster, but also easier to use. Each box can also contain tabs.
We wanted the current balance of the users to be always visible, therefore we placed a bar chart displaying that information in the header.
All settings (except where you can customize the statistics) are on a separate page.
We placed the expense tool on the upper left of the page, since we think, that this will be the tool that is used most. The statistics tool is placed on the bottom of the page to take advantage of the full page width.
We tried to make the page more usable by automatically focussing on the most appropriate input field. As soon as the user is logged in, the focus is automatically on the field where you can enter the amount of an expense. On the other hand, if you create a new shoppinglist, the focus is on the field, where you can add new items to the shoppinglist.



Logged in as Jonas Luder (Logout)

EN | DE

wg_2

2011-10-19

Account: Haushalt

Name: Name

visible for: me

Create more accounts for your flat share.

@wgfinance · 3 followers

6 likes. Sign Up to see what your friends like.

Figure 3.1: The new wgfinance.

Chapter 4

Security and Data Privacy

When developing a service for today's World Wide Web, one should be quite cautious, as there are many, partly automated, adversaries that try to exploit various vulnerabilites; or, as they've put it in [4]:

The Internet can be a scary place.

In this chapter we illustrate how security issues are handled in the Django framework and wgfinance, respectively. Generally speaking, there are two different perspectives we have to focus on, namely the perspective of the client and the server's.

4.1 The Client's Perspective

A client's main concern, if even any, is usually the protection of the data he or she provided. To account for this we considered the following:

4.1.1 HTTPS

In order to authenticate the server to the client on one hand, and to encrypt the traffic on the other, we deployed HTTPS, hence certificates. With that we can establish a secret channel between the server and the user and hence, the client need not worry about his data being eavesdropped.

4.1.2 API Security

The API is typically used for external applications, that want to use the capabilities of wgfinance and include it within this application. Since we cannot guarantee the well-functioning of those applications, it would not make sense to use the username and password combination that we use within wgfinance itself for authentication. We took the decision to rather use a username and token combination. For allowing an external application access to wgfinance, the user has to generate a token within wgfinance and then use it to connect from within the exernal application.

This is particularly useful in the case where the external applications are run on portable devices. Upon loss of such a device the user can just simply deactivate the token within wgfinance to prevent further access by the external application. In addition the password itself has not been compromised, since it was never stored on the lost device.

4.2 The Server's Perspective

From the server's point of view the security challenge is manifold. According to [4, p. 341] the most important paradigm concerning how to deal with the threat can be stated as follows:

Never - under any circumstances - trust data from the browser.

Having this paradigm in mind, let's look at some frequent exploits.

4.2.1 SQL Injection

In an SQL injection exploit an attacker typically makes use of form parameters, i.e. GET or POST parameter, that are put into a SQL query without being properly escaped. Since an SQL query is directly executed on the database, this type of exploit has quite a big impact.

Thanks to the sophisticated Django database API all parameters given to the API functions are escaped automatically. The only point where we did have to pay special attention is obviously in cases where we did deliberately omit the database API. Since this happens only at one point in the code, we were able to ensure that parameters are being escaped.

4.2.2 Cross-site Scripting

Another typical attack, which again makes use of browser data not being correctly escaped, is known as cross-site scripting, or XSS for short. In this case GET or POST parameters are poisoned with possibly malicious HTML tags, such as iframes et al, that are then directly put onto the returned webpage.

In Django however the mighty template engine generally escapes every variable that it is given.

4.2.3 Directory Traversal

As on one hand we do not explicitly open files in our code, and on the other hand all urls that are requested by a client are matched against regular expressions, there is virtually no risk of directory traversal.

4.2.4 Session Hijacking

Since we use HTTPS, as already mentioned above, a secret channel between the client and the server can be established, which eliminates the risk of a session being hijacked. However one is still vulnerable against Man-in-the-Browser attacks.

4.2.5 API

Regarding the API we do have to be very paranoid, as any possibly malicious application can access it. Therefore we observe that 1. we have to always comply with the above paradigm and hence 2. design the API calls in a very strict and clearly defined manner.

Chapter 5

Competitors and Rivals

Obviously, there are dozens, if not hundreds of other websites that provide functionalities similar to wgfinance. Below we analyzed three of them and provide comparisons to wgfinance.

5.1 splitabill

splitabill.com[1] essentially provides the functionality we call the expense tool. However, in splitabill every expense is looked at separately and therefore it is not efficiently possible to look at statistics and every expense needs to be settled separately. This may be suitable to settle occasional expenses with varying partners, but it is a very inefficient way of keeping track of regular expenses with the same partners, as it would be the case in a flat share. From a flat share's point of view wgfinance would better suit its needs.

5.2 iou.ch

iou.ch[2], which has been developed by fellow ETH students, has a very sophisticated way to split bills. It is for instance possible to split bills percentaged, whereas wgfinance is only able to split them equally among the flat mates. It has a very doodle-like user interface to create the so called pots. In contrast to the multi-level account scheme of wgfinance, iou.ch supports only a single account called a pot to register all expenses. Therefore iou.ch can only provide very limited statistics.

[1] `https://splitabill.com`

[2] `http://www.iou.ch`

5.3 abrechnung-wg.de

The tool that matches wgfinance best is abrechnung-wg.de[3]. Many features of the two tools are similar. However abrechnung-wg.de provides additional features, namely a budget planning tool and automated regular expenses. wgfinance on the other hand provides splitting of expenses, which abrechnung-wg.de doesn't. Furthermore the statistics of wgfinance seem more sophisticated than those of abrechnung-wg.de, since they are able to show multiple accounts within the same chart. Generally wgfinance seem to be laid out more clearly, due to the fact that abrechnung-wg.de uses more pages to provide the same features.

[3]`http://abrechnung-wg.de`

Chapter 6

Conclusion

Taking the above comments on our rivals into account, we observe that in some cases wgfinance is in an inferior position and in other cases superior. We can therfore conclude that wgfinance provides a competitive service, but does not provide revolutionary new functions.

6.1 wgfinance Future

As mentioned on various occations above, there are numerous features we already came up with. In the following we provide a list as well as a description of those features as an inspiration for future development of wgfinance. An implementation of the features below would turn wgfinance into an even more competitive tool.

- **Budget**
  A budget tool for planning and monitoring accounts of a flat share is probably the feature that has to be implemented next. This feature would provide a simple form where expense limits for certain accounts could be specified. Furthermore the statistics would have to provide an appropriate visualization of budget states.
- **Export**
  Even though all expenses can be examined in the statistics, there might be the need for a hardcopy. For instance the hardcopy could be needed for archiving, or even as a backup. An export functionality of a given statistics would therefore be a very handy feature.
- **Website for Mobile Clients**
  Generally the time between an expense itself and the its registration in wgfinance has to be kept small. It would therefore be desirable that expenses are registered almost in real-time, hence on mobile clients. As the display of those devices is usually small, a special website is required for these devices.

- **Autocomplete**
  A small but useful feature would be an autocomplete function for the comment in the expense tool, as well as for items in the shoppinglist. This feature would provide suggestions based on previous entries while typing.

- **API Extension**
  Currently the above specified API provides access only to the wgfinance shoppinglist feature. It is desirable that the API is extended as to provide the whole wgfinance functionality, especially registering expenses.

- **Embedding `http://aktionis.ch`**
  By embedding the website aktionis.ch, which provides an overview of sales promotions by multiple companies such as Migros, Coop or Denner, the shoppinglist feature could be made even more sophisticated. Given items of a shoppinglist could be compared to the promotions, and suggestions could be given regarding the store at which the item could be bought.

- **Social Sharing**
  In order to promote wgfinance and to attract more users the integration of wgfinance into social networking sites has to be deepened. One could an imagination such that a given facebook user can share that he or she is using wgfinance in a simple manner.

There is currently already an ongoing project: A flat mate of Lorenz is developing a shoppinglist app for android mobile phones, that makes use of the wgfinance API.

Appendix A

Developer's Notes

A.1 Application Programming Interface

A.1.1 General Remarks

- All URLs have to be encoded as defined in RFC1738[1].
- All strings are encoded in UTF-8.
- Arguments like `<api_token>` have to be replaced by the variable.
- Arguments like `[person_id]` have to be replaced as well, but are optional.
- The root tag of every responce is the `api_v0` tag.
- If the status is not ok, this tag is always empty:
  - If the request url does not match any pattern shown below:

    ```
    <api_v0 status="request_invalid" />
    ```
  - If the token is invalid:

    ```
    <api_v0 status="auth_token_invalid" />
    ```

[1] http://www.rfc-editor.org/rfc/rfc1738.txt

A.1.2 Function Reference

- `wg_listing_add`
  - success

    ```
    http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
       wg_listing/add/?name=a%20new%20list

    <api_v0 status="ok" wg_id="1" person="LK" >
    <wg_listing_add status="ok" >
    <listing id="12" v_id="24"
    modified="2011-01-01 20:41:50"
    modified_by="LK"
    name="a new list" />
    </wg_listing_add>
    </api_v0>
    ```

  - If the name parameter is omitted:

    ```
    http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
       wg_listing/add/
    <api_v0 status="ok" wg_id="" person="" >
    <wg_listing_add status="name_missing" />
    </api_v0>
    ```

- `wg_listing_delete`
  - success

    ```
    http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
       wg_listing/delete/9/

    <api_v0 status="ok" wg_id="1" person="LK" >
    <wg_listing_delete status="ok" id="9" />
    </api_v0>
    ```

  - If the listing does not exist or does not belong to the person or the flat share of the token:

    ```
    <api_v0 status="ok" wg_id="1" person="LK" >
    <wg_listing_delete status="invalid_listing_id" />
    </api_v0>
    ```

- `wg_listing_get`

  -
    ```
    http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
       wg_listing/get/
    ```

```
<api_v0 status="ok" wg_id="1" person="LK" >
<wg_listing_get status="ok"  >
<listing id="1" v_id="3"
modified="2010-12-31 17:03:29"
modified_by="LK"
name="test 0" />
<listing id="2" v_id="9"
modified="2011-01-01 14:57:54"
modified_by="LK"
name="second listing" />
<listing id="10" v_id="21"
modified="2011-01-01 15:42:14"
modified_by="LK"
name="another listing" />
</wg_listing_get>
</api_v0>
```

– http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
   wg_listing/get/?items=true&listing_v_id_gt=1&item_v_id_gt=10

```
<api_v0 status="ok" wg_id="1" person="LK" >
<wg_listing_get status="ok" listing_v_id_gt="1" item_v_id_gt="10" >
<listing id="1" v_id="23"
modified="2011-01-01 20:36:47"
modified_by="LK"
name="new name" >
<item id="1" v_id="20"
modified="2011-01-01 20:50:15"
modified_by="LK"
content="modified content" />
</listing>
<listing id="2" v_id="9"
modified="2011-01-01 14:57:54"
modified_by="LK"
name="second listing" >
<item id="10" v_id="11"
modified="2011-01-01 14:13:56"
modified_by="LK"
content="third thing" />
<item id="11" v_id="12"
modified="2011-01-01 14:14:10"
modified_by="LK"
content="third thing" />
<item id="13" v_id="14"
```

```
modified="2011-01-01 14:24:02"
modified_by="LK"
content="&quot;another test&quot;" />
</listing>
<listing id="10" v_id="21"
modified="2011-01-01 15:42:14"
modified_by="LK"
name="another listing" />
<listing id="12" v_id="24"
modified="2011-01-01 20:41:50"
modified_by="LK"
name="a new list" />
</wg_listing_get>
</api_v0>
```

  - `items=true`
    If the get parameter `items == 'true'`, the items are returned.
  - `listing_v_id_gt=n`
    If the get parameter `listing_v_id_gt` is set, only listings with a `listing_version_id` > `listing_v_id_gt` are returned.
  - `item_v_id_gt=n`
    If the get parameter `item_v_id_gt` is set, only items with a `item_version_id` > `item_v_id_gt` are returned.

- `wg_listing_modify`
  - `http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \`
    `wg_listing/modify/1/?name=new%20name`

```
<api_v0 status="ok" wg_id="1" person="LK" >
<wg_listing_modify status="ok" >
<listing id="1" v_id="23"
modified="2011-01-01 20:36:47"
modified_by="LK"
name="new name" />
</wg_listing_modify>
</api_v0>
```

  - No name given:

```
http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
   wg_listing/modify/1/

<api_v0 status="ok" wg_id="1" person="LK" >
<wg_listing_modify status="name_missing" />
</api_v0>
```

  - If the listing does not exist or does not belong to the person or the flat share of the token:

    ```
    http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
       wg_listing/modify/42/?name=foo

    <api_v0 status="ok" wg_id="1" person="LK" >
    <wg_listing_modify status="invalid_listing_id" />
    </api_v0>
    ```

- `item_add`

  - success:

    ```
    http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/item/ \
       add/1/?content=new%20item

    <api_v0 status="ok" wg_id="1" person="LK" >
    <item_add status="ok" listing_id="1">
    <item id="15" v_id="18"
    modified="2011-01-01 20:44:57"
    modified_by="LK" content="new item" />
    </item_add>
    </api_v0>
    ```

  - If the listing does not exist or does not belong to the person or the flat share of the token:

    ```
    http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/item/ \
       add/1/?content=new%20item

    <api_v0 status="ok" wg_id="1" person="LK" >
    <item_add status="invalid_listing_id" />
    </api_v0>
    ```

- `item_delete`

  - success

    ```
    http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/item/ \
       delete/15/

    <api_v0 status="ok" wg_id="1" person="LK" >
    <item_delete status="ok" id="15" />
    </api_v0>
    ```

  - If the item does not exist or does not belong to the person or flat share of the token:

```
<api_v0 status="ok" wg_id="1" person="LK" >
<item_delete status="invalid_item_id" />
</api_v0>
```

- `item_get`
  - success

```
http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
   item/get/1/

<api_v0 status="ok" wg_id="1" person="LK" >
<item_get status="ok">
<listing id="1" v_id="23"
modified="2011-01-01 20:36:47"
modified_by="LK"
name="new name" >
<item id="1" v_id="20"
modified="2011-01-01 20:50:15"
modified_by="LK"
content="modified content" />
<item id="2" v_id="2"
modified="2010-12-31 14:51:18"
modified_by="LK"
content="foo" />
<item id="3" v_id="3"
modified="2010-12-31 14:51:26"
modified_by="LK"
content="another item" />
<item id="4" v_id="4"
modified="2010-12-31 14:56:43"
modified_by="LK"
content="Milch" />
<item id="5" v_id="6"
modified="2010-12-31 14:59:21"
modified_by="LK"
content="Kuchen" />
</listing>
</item_get>
</api_v0>
```

  - If the Listing does not exist or does not belong to the person or the flat share of the token:

```
http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
   item/get/1/
```

```
<api_v0 status="ok" wg_id="1" person="LK" >
<item_get status="invalid_listing_id" />
</api_v0>
```

  - `http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \`
    `item/get/1/?item_v_id_gt=4`

```
<api_v0 status="ok" wg_id="1" person="LK" >
<item_get status="ok" item_v_id_gt="4" >
<listing id="1" v_id="23"
modified="2011-01-01 20:36:47"
modified_by="LK"
name="new name" >
<item id="1" v_id="20"
modified="2011-01-01 20:50:15"
modified_by="LK"
content="modified content" />
<item id="5" v_id="6"
modified="2010-12-31 14:59:21"
modified_by="LK"
content="Kuchen" />
</listing>
</item_get>
</api_v0>
```

  - `item_v_id_gt=n`
    If the get parameter `item_v_id_gt` is set, only items with a `item_version_id` > `item_v_id_gt` are returned.

- `item_modify`
  - success

```
http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
   item/modify/1/?content=modified%20content

<api_v0 status="ok" wg_id="1" person="LK" >
<item_modify status="ok" listing_id="">
<item id="1" v_id="19"
modified="2011-01-01 20:49:24"
modified_by="LK" content="modified content" />
</item_modify>
</api_v0>
```

  - If the item does not exist or does not belong to the person or flat share of the token:

```
http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
   item/modify/1/?content=modified%20content

<api_v0 status="ok" wg_id="1" person="LK" >
<item_modify status="invalid_item_id" />
</api_v0>
```

–
```
http://localhost:8000/api_v0/9ebd6c2bd3a5234bd81a817b23b9a099/ \
item/modify/1/

<api_v0 status="ok" wg_id="1" person="LK" >
<item_modify status="content_missing" />
</api_v0>
```

References

[1] Michael Bowers. *Pro CSS and HTML Design Patterns.* Apress, 1 edition, April 2007. ISBN 9781590598047.

[2] Django Software Foundation. The official django documentation. https://docs.djangoproject.com/en/1.3/, October 2011.

[3] Magnus Lie Hetland. *Beginning Python: From Novice to Professional.* Apress, 2 edition, June 2010. ISBN 9781590599822.

[4] Adrian Holovaty and Jacob Kaplan-Moss. *The Definitive Guide to Django: Web Development Done Right.* Apress, 2 edition, July 2009. ISBN 9781430219361.

[5] Allan Jardine. Datatables examples. http://datatables.net/examples/, October 2011.

[6] jqPlot. jqplot usage documentation. http://www.jqplot.com/docs/files/usage-txt.html, October 2011.

[7] The jQuery Project. jquery documentation. http://docs.jquery.com, October 2011.

[8] Mark Lutz. *Learning Python.* O'Reilly, 4 edition, October 2009. ISBN 9780596158064.

[9] Oracle. Mysql 5.5 reference manual. http://dev.mysql.com/doc/refman/5.5/en/, October 2011.