Master Thesis

"Ethire" - a system to support the faculty hiring process

Author(s):
Loeu, Matthias

Publication Date:
2009

Permanent Link:
https://doi.org/10.3929/ethz-a-005816776

Rights / License:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection. For more information please consult the Terms of use.
“Ethire” – A system to support the faculty hiring process

Master Thesis

By: Matthias Loeu
Supervised by: Marco Piccioni
Prof. Bertrand Meyer

Student Number: 03-928-207
Abstract

Hiring new faculty at ETH is a complex and sometimes long process; some of its aspects are tedious, and a lot of documents have to be produced and maintained.

Upon examining the work flow we identified areas which benefit most from software support. These were handling the applications by the candidates, the phone book and the first meeting.

The software support was built using techniques from Software Engineering. It has a three-tier architecture, and is highly scalable. The top tier is a PHP application and handles user input. The middle tier is an Eiffel application dealing with business logic and most of the data. The bottom tier handles data storage.

The administrative staff is very satisfied with the end result and will use the software in the future.

This report covers the scope of my work and the achieved results as well as the problems I have encountered and possible future work that can be done based on my results.
# Table of Contents

1 Introduction....................................................................................................................5  
   1.1 Organization of this document................................................................................5  
2 Process Analysis.............................................................................................................5  
   2.1 Current process.......................................................................................................5  
      2.1.1 Work flow.......................................................................................................5  
      2.1.2 Graphical Representation..............................................................................6  
      2.1.3 Additional Information................................................................................7  
   2.2 Areas which benefit from software support.........................................................8  
   2.3 Existing Solutions.................................................................................................8  
3 Software Support..........................................................................................................8  
   3.1 User Interface........................................................................................................9  
   3.2 Architecture.........................................................................................................10  
   3.3 Front-end.............................................................................................................10  
      3.3.1 Work flow.......................................................................................................10  
      3.3.2 Technologies..................................................................................................11  
   3.4 Back end...............................................................................................................11  
      3.4.1 Work flow.......................................................................................................11  
      3.4.2 Technologies..................................................................................................11  
   3.5 Database..............................................................................................................12  
   3.6 Similarities to other software..............................................................................13  
4 Problems/Challenges....................................................................................................13  
5 Future Work..................................................................................................................13  
6 Conclusion....................................................................................................................14  
7 References...................................................................................................................14  
8 Appendices..................................................................................................................15  
   8.1 Terminology..........................................................................................................16  
   8.2 User Guide............................................................................................................18  
   8.3 Developer Guide.................................................................................................23  
   8.4 Installation Guide.................................................................................................27
Acknowledgments

First I would like to thank Marco Piccioni for supervising my master thesis. I really appreciated the amount of freedom he allowed me in designing my work, all the patience needed as well as all the input I received.

My gratitude also goes to Prof. Bertrand Meyer and Dr. Thomas Eichenberger for giving me the unique opportunity to write this thesis in the field of hiring faculty members.

Special thanks to Franziska Brunner and Simone Jacober from the administrative staff, both of which I am very honored to have worked with. Without their support and contributions this master thesis would not have come true.

Finally, a great thank you to Conny for all her encouragement and all the reminders during this time as well as for proofreading this report.
1 Introduction

Hiring new faculty at ETH is a complex and sometimes very long process; some of its aspects are tedious, and a lot of documents have to be produced and maintained. At the moment no off-the-shelf software is directly applicable to handle them optimally.

The goal of this project was to study the process in detail to see what parts could benefit from software support, and to build this support.

1.1 Organization of this document

Section 2 covers the analysis of the process, section 3 describes the overall architecture of the software support. Section 4 covers problems I encountered during the work. Future work is outlined in section 5, and I conclude in section 6.

The appendices give a definition of the terminology used and include a user, developer and installation guide for the software.

2 Process Analysis

2.1 Current process

Every step is done manually. At the beginning the new position is advertised. Applications come in both by E-Mail and paper mail. Sometimes they contain a substantial amount of data. After the deadline the candidates go through multiple steps, with most of them dropping out at the beginning. The president negotiates the position with with the few candidates remaining at the end.

The process described here is only one part in a greater hiring process [1].

Below is a more formal list of steps that are performed. See also the appendix (section 8.1) for a glossary.

2.1.1 Work flow

The work flow of a hiring procedure is as follows:

1. Develop and approve the profile paper
2. Advertise the position in journals, newspapers, on web pages, etc.
3. Gather all the paper applications, send confirmation of receipt to all the candidates
4. Prepare the phone book and send it to the committee members (by paper mail)
5. Send additional dossiers to the committee members by E-Mail
6. Have the first meeting. Reject all but four to eight people. Three to six people are assigned as A candidates and one to two as B candidates

1. Send C candidates letter of rejection
2. Write the minutes and send them by E-Mail to committee members
7. Invite the A candidates to the symposium and the interview, organized by the contact person
8. Have the second meeting. This consists of
   1. Interview with A candidates
   2. Split the A candidates into A+ and A- candidates
   3. Rank the A+ candidates.
   4. Write minutes (with ranking) and send them by E-Mail to the committee chair
9. Prepare Memo 3 for the president, then send it to him
10. Have final meeting consisting of president, chair and staff
    1. Decide on the top candidate (usually ranked first in the second meeting) and invite him for negotiations
    2. Send A- candidates a letter informing them about the fact that negotiations with another candidate are ongoing and that the president may get back to them
    3. Send committee members a document consisting of the minutes of the second meeting and the presidents decision whom to invite for negotiations
11. Negotiate with the top candidate
12. Assuming the negotiations were successful, hire that candidate and send all the remaining candidates letters of rejection
13. Otherwise find a new top candidate (usually ranked second) and negotiate with this candidate, repeating the above three steps

2.1.2 Graphical Representation
Here is a graphical representation of the above:
2.1.3 Additional Information

The following figures are a rough estimate of the order of magnitude of the hiring procedure with respect to time length, number of applications received and number of concurrent active open positions. It illustrates the workload of the administrative staff.

<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall time from start to end</td>
<td>9 months at the least</td>
</tr>
<tr>
<td>Time from publishing the position advertisement to the formal deadline for applying</td>
<td>about 3 months</td>
</tr>
<tr>
<td>First meeting</td>
<td>4-6 weeks after deadline</td>
</tr>
<tr>
<td>Send out phone book</td>
<td>roughly 3 weeks before first meeting</td>
</tr>
<tr>
<td>Candidates who apply after the deadline</td>
<td>roughly 5%</td>
</tr>
<tr>
<td>Number of candidates who apply</td>
<td>up to 200 candidates</td>
</tr>
</tbody>
</table>
2.2 Areas which benefit from software support

The areas which benefit most from software support are those in which many people are involved and where a great number of documents have to be processed. These are:

- Application by the candidate
- Handling the phone book
- Separating candidates to A, B and C candidates in the first meeting

Although it would be desirable to use one system for the whole work flow, due to time and manpower constraints we had to focus on just these three points. Other areas are touched upon by the software, but were not the main concern. These other areas, as well as special cases, can be handled manually (just as it is done right now). Since not many people are involved, the manual labor can be kept at a minimum.

2.3 Existing Solutions

There are no existing solutions which can be used on the spot for this procedure.

There are, however, tools which can be used to model this process. One example are the services of SAP [2]. Others include all the custom software packages any consulting company can provide. The drawback to all these solutions are the fact that a lot of work would have needed to be done for adapting those solutions to the current process. In addition to the manpower needed there is also a large cost in software licenses – the solution presented here is Open Source.

3 Software Support

The software presented here is called Ethire, its project page is available on this address: [3]. A running demo-system can be found here: [4].
3.1 User Interface

To the user, Ethire is a web-application. The reason for that is, it should be easily accessible and leave the least amount of traces on the computers of its users. We can assume that everyone has a working browser installed. All documents which are served to the user are either

- Standard XHTML documents [5] for the browser, or
- PDF documents [6] for printing

These documents are standardized and widely understood by contemporary computers. In addition, a small number of items are served as CSS (styles), PNG and JPEG (images).

The user interface follows the Corporate Design of ETH [7]. Building upon the existing templates, they were updated considerably to make them comply fully to the W3C standard [5].
3.2 Architecture

Ethire is built using a classic three-tier architecture. This allows the use of modular software with a well defined interface. Advantages are the ability to run each module on different servers for scalability, the ability to run multiple instances of the Front- and Back-end for even more scalability, and the ability to upgrade each module individually. The Front-end corresponds to the Presentation tier, the Back-end to the Application Logic tier. The Databases corresponds to the Data tier. The Browser illustrated is an important part of the user interface, but not part of the actual software.

3.3 Front-end

3.3.1 Work flow

The Front-end is built on top of a web server. It receives requests from the user and processes them. It

1. Finds the class and method to invoke. Then it validates and normalize all input parameters
2. Checks if there are sufficient privileges to execute the request, if not, it redirects the user to the homepage or the last visited page
3. Executes the request and invokes the Back-end as needed
4. Builds result structure
5. Populates XHTML page with content and sends it to the user

There are some exceptions to this: For example, PDF documents are created once and cached on the file-system. If the user requests other documents or images, they are read and sent to the user directly without querying the Back-end. Finally, most requests for
modification of data are carried out, but then the user is sent back to the previous page (where he can see the modified data).

### 3.3.2 Technologies

The technology used is PHP [9]. The reason is the fact that we started with the user interface and built the Front-end iteratively from there. Also, due to the nature of weak typing and dynamic data types changes can be incorporated very rapidly. The PHP Front-end can be considered a prototype which, with time, may be replaced with something more strict once the majority of change requests are done (see also Section 5). Additionally, we use dynamic binding (polymorphism) to choose which Back-end to use. Reflection is used to find and invoke the appropriate class and method.

The Login is done externally via Identity Providers. Most users will probably use their ETH account, however users from outside ETH have the possibility to use their Login data from many providers like Facebook, Xing, MSN or many others. The largest benefit is the fact that users do not have to remember yet another password. Do note that while users enter their password, the software does **not** store any passwords whatsoever.

### 3.4 Back end

#### 3.4.1 Work flow

The Back-end is built on top of a web-server like the Front-end. Its internal architecture is very similar to the Front-end, as it processes requests from the user (which in this case is the Front-end). It

1. Validates and normalizes all input parameters. It then finds the class and feature to invoke.
2. Depending on the feature, builds one or more SQL statements
3. Queries the Database
4. Builds result set and serializes it
5. Sends result back to the user (the Front-end)

#### 3.4.2 Technologies

The technology used here is Eiffel [8], with its base libraries EiffelWeb and EiffelStore. As the Back-end is the Application Server and contains a lot of *business logic*, the use of this technology was more reasonable.
Communication with the Front-end is done via HTTP and PHP serialization using a fixed API. Other encodings are possible, however in this case both the Front- as well as the Back-end would have to be changed considerably. Also, different encodings may be used in parallel then.

Communication with the Database is done via ODBC and its accompanying library EiffelStore.

### 3.5 Database

The Database holds all relevant data except documents. Those are stored as files on the server for performance reasons. There are actually two databases, one for the Front-end and one for the Back-end.

The Front-end Database (Database 1 in Illustration 2) holds minor data, such as the language of the current user, login data, connection data and so on. All database table definitions are stored in the Front-end. Specifically, in the install-method of each database connection class. See also the developer guide for more information.

The Back-end Database (Database 2 in Illustration 2) holds all application related data. Those are, names of candidates, committee members, deadlines, documents and so on.
3.6 Similarities to other software

The most striking similarity this application has is the one to Origo [10]. The Origo architecture is indeed used as an inspiration to the design of the current architecture: Both are built using the three-tier architecture.

The Front-End is equivalent to Origo-Web. Both are PHP applications. In the case of Origo-Web, it is built as Drupal [11] modules. The Ethire-related Front-End on the other hand is one module in a custom PHP-Framework.

The equivalent to the Back-end is Origo-Core. Both use Eiffel. In the case of Origo-Core the Gobo libraries [12] are used, while the Back-end uses EiffelWeb and EiffelStore. The communication with the Front-end (Origo-Web) is done via XML-RPC, which would have been a possibility with Ethire as well (see section 5 for other alternatives).

Origo as well as Ethire use MySQL as a database, both in a very similar way as there are two databases, one for the Front-end and one for the Back-end.

4 Problems/Challenges

The most notable problem we have encountered is the lack of time. Only the most immediate aspects of supporting the hiring process were considered. If more time were available, a more thorough support could have been provided. See also section 5 for possible areas where more time would have been beneficial.

The technological and methodological constraints, and observance of strict constraints of confidentiality posed some challenge. Also the close cooperation with the administrative units involved – while enjoyable – posed some challenges in translating the language from the problem domain to the solution domain and vice versa.

Due to the different technologies used, correct communication between each system was crucial, this posed many challenges, but ultimately were a small problem.

Many change requests were made at a very late state, after much of the work had been done already. Finding time to incorporate them was difficult.

The PDF generator continues to be somewhat slow when joining pre-existing documents. Especially in the case where many hundred pages have to be read, parsed and recreated.

5 Future Work

Possible advances can be made in the following fields:

• A real life case study may be done of one position using the software support presented here. This would include surveying the people involved, asserting the
actual use of various parts of the software and writing recommendation as to where improvements can be made.

- The Front-end can be replaced with one using a strongly statically typed language such as Eiffel (for example with EiffelWeb). As the Front-end matures in the future, it would be easier to maintain in the long run because of its strict adherence to Object oriented methodology.

- communication methods may be replaced or added between the Front- and Back-end. Possible technologies are
  - Standard JSON encoded data
  - Standard Eiffel Object Serialization
  - Custom XML or XML-RPC

- OpenID [13] and/or Shibboleth [14] may be integrated for authentication purposes. Due to the nature of the existing authentication process, this should be a relatively easy task.

- Replace the PDF renderer with something more scalable and powerful.

- Design and implement the ability to use all the different ETH letterheads/templates in PDF documents.

6 Conclusion

We have presented the overall procedure and reasonable areas where we can improve the procedure. In addition the application has been iteratively tested by the administrative staff and seems pretty stable.

Even though there were some problems or challenges, these were not major. And there still remains some work – especially a dress rehearsal – the application is ready for actual use! The administrative staff is already working very hard to get acquainted with the application.

I personally have very much enjoyed working with this project. Taking into account what has been done, the project can be considered a success.

7 References

8 Appendices

The appendices hold more detailed data than which was presented in the main body of this report. They serve as a reference.

The terminology, user, developer and install guides can be found (and updated) on the project pages [3].
8.1 Terminology

This appendix holds the domain specific terminology used throughout this document, with a short explanation.

Candidates:

A Candidate: Candidate who is invited for the symposium and interview.

A- Candidate: Candidate who has been an A candidate, but does not make it to the ranked list.

A+ Candidate: Candidate who has been an A candidate and who also makes it to the ranked list.

B Candidate: Candidate who, though not invited for the interview, is considered as a reserve candidate after the first meeting. After the president's decision whom to invite for negotiations, the B candidate can be either promoted to A or demoted to C.

C Candidate: Candidate who is rejected in the first meeting. After the first meeting, a letter of rejection will be sent.

Documents:

Dossier: Collection of documents about one specific candidate.

Memo 3: The package which contains the position advertisement, the list of committee members, the list of all applicants (without their dossiers), all minutes and the dossiers from all A+ candidates.

Phone book: Compilation of all dossiers prior to the first meeting. Additionally, it also contains

- an information letter (what the committee members are supposed to do)
- a list of the committee members & candidates
- information about the first meeting
- a confidentiality statement
- the duties of committee members.

Profile Paper: Document meant to present information about the advertised position. It contains (a draft of) the position advertisement, a list of the "Committee members", financial information (available budget), a list of potential candidates and other information.

Events:

First Meeting: Meeting of the entire committee to discuss all the candidates who have applied. The candidates will be divided into A, B and C Candidates.
**Second Meeting:** Also known as the colloquium. Meeting of the entire committee to discuss the A Candidates, split them to A- and A+ candidates and to rank the A+ candidates.

**Final Meeting:** Meeting of the department head, the president and the committee chair to discuss the (then already ranked) A+ Candidates. They also elect the top candidate.

**People:**

**Candidate:** The person who applies for a faculty position at ETH. (See above)

**Committee:** Group of people that participate in the first and second meeting. It consists of:

- Committee chair
- One person from the *Office for Faculty Affairs*
- Three to six professors from the department, among which are
  - The Department Head
  - The contact person
- One to three professors from other departments
- One member of the non-professorial staff
- One member representing EPFL
- One member representing the University of Zurich
- One student representative
- Other members from outside ETH, representing other (usually foreign) universities, the canton or the industry

**Committee Chair:** The head of the committee. He is an ETH professor coming from another department with respect to the one for which the position is advertised. He represents the president.

**Contact Person:** The person that is in charge of:

- Keeping contact with the outside world, for example by answering to questions about the advertised position
- Organizing the colloquium

**ETH President:** The person who makes all the decisions.

**Process Administrator:** Persons from the *Office of Faculty Affairs*, dealing with administrative duties

**Staff:** People from the *Office of Faculty Affairs*, involved with decision making
8.2 User Guide

This appendix contains a copy of the user guide.

Everyone:

Language:

You can change the language by clicking on the appropriate link on the top right of every page.

Help:

There is a general help area within this application. To access it, click on Help on the upper right corner of every page (left of the language links).

Candidates:

Applying:

You have two possibilities to apply for any given position. One is the use of the apply-link on the right hand side. In this case you'll have to choose the position as a first step. The other is browsing through the open positions on the main area. If you apply via one of those links, the first step is done automatically and you start with step 2.

A more step by step guide can be found on the help pages.

Committee Members:

Basic work flow:

1. Log In
2. Select the appropriate position you're working on. (Usually there is just one anyway.)
3. You'll now see an overview about the position.
4. Near the bottom you can find documents which you can download.
5. For the list of candidates who have applied thus far, on the right hand side there is a box called Actions, there is a link for List of candidates

   1. In the beginning all candidates have either the status new or nothing. Click on More Information to view the candidates' details.

   2. Now you can rate the candidate, add notes (both see below) or download the candidates' entire dossier (if you want to read the same information without the computer). You can also download individual documents if you so want.

6. In the Phone Book section you can download the General Information Letter and a compilation of all candidate dossiers. These documents are intended to be printed out and are – more or less – the same as for any other hiring procedure not using this application.
7. On the first meeting you will be dividing the candidates to A, B and C. The actual assignment within the application will be done by the administrator.

8. After the first meeting you can only see those candidates who are still interesting. In order to see the rest, click on View All Candidates. In the event of new candidates who apply even after the first meeting, they will show up on the main candidate list.

Candidate Lists:
There are basically two lists. The first (and main) one displays only the candidates who are interesting. These are basically everyone except those denoted by B or C. The other one displays every candidate. The candidate list you can download from the phone book contains all the candidates.

Log In (Authentication):
You will need to login in order to use the software as a committee member. For that purpose you can use your existing ETH credentials (or, in the case of external members, those of any other supported identity provider). Be sure to select the correct Identity Provider! If successfully logged in, you will be redirected to the previous page you were on.

Notes:
There are two types of notes. The first being the public notes. Every committee member is able to see and edit these notes. They are intended to be a public debating area about the candidate. Note that the candidate himself can not view the notes about him.

The second type are the private notes. These notes only you, the committee's chair and the administrator can view. Here you can for example write your thoughts about a candidate which are confidential with respect to other committee members but are still important.

If you want to write down notes which you and only you can view, please do so without the use of this application – for example in a text document or on a sheet of paper. This ensures a maximum of privacy (the things you write can never be published by the application even by accident).

Personal Information:
Once you have logged in, you can see your name on the top left corner. If you click on it, you go to a page with your basic user information. You can edit these with the links on the right hand side. To get back you can click on Home or on the title on the top middle of the page.

Phone Book:
The phone book is a compilation of all candidate dossiers (as well as some other information). You can download the phone book at any time you want. It will (obviously) just contain the candidates who have applied up until that point.
If something changes and you want to re-download the phone book you of course you have that possibility. However, you can also download a reduced copy of the phone book with only those candidates who are new (or have changed some details about them in the meantime).

**Ratings:**

You can rate all candidates. These ratings are strictly informal only and should help quickly categorize the candidates.

In order to view ratings of other committee members, you first have to give a rating yourself. The reason being that the existing ratings should in not influence your own rating.

After you have rated, you can still change your rating by clicking on the exact same link. However this is not recommended.

**Administrators:**

**Assigning Administrators:**

You can add or remove users as administrators on the appropriate page. Do note that new administrators will need a user account (see below) first. Also, you can't remove yourself as an administrator.

**Candidate Status:**

You can change the status of a candidate on the page where his details are displayed (go to position → list of candidates → more information). On the right hand side you'll find the appropriate link.

**Candidate Status Types:**

The types refer to the status the candidates can have. Once set up, they should rarely change.

The filtered-column indicates whether they are to be shown on the candidate list. (The other lists show all candidates regardless of status.) The color indicates the rows color in the candidate list. The default status refers to the status a candidate receives just after he has applied for a position.

**Contact Information for Committee Members:**

Every committee member has an E-Mail address and some contact information. You can edit this info by clicking on a position, where you get the list of committee members. There is a link, More Information, which brings you to a new page where you can edit that users info on the right.

Note that committee members can only edit their own contact info.

**Create a new position:**

1. Select the appropriate link on the right side. Fill in the details, not everything is required, just the name of the position is necessary.
2. Create users for committee members who are not (yet) in the system. The appropriate link is under further administrative actions.

3. Go to the newly created position and add the committee members. Also select the Head and Chair.


Create a new user:
In order to get committee members into the system you have to create a user account for them. You will need only some basic information, important is the user name and organization he will use in order to log in. If the committee member is not from ETH (or if you don't know their log in name) they may register as a user themselves.

Edit candidate information:
This can be done on the candidate information page. Note that if you can not edit a document, just add a new one.

Identity providers:
Here you can change the organizations a user can log in with. Note that you can't disable the organization which you have used to log in yourself and also disabling those where log in data exists can lead to unpredictable results (you can view the number of users who use said log in provider under Info).

New candidates:
After a candidate has applied he will show up on the candidate list of the appropriate position with the status “new”. You can then review that information, send a confirmation E-Mail to the candidate and change his status to “-”.

There is currently no page which displays only the new candidates.

PDF Documents and Templates:
All PDF Documents Letters which will are generated assume to be printed on paper with an ETH-letterhead.

Position Categories:
Here you can change the names of the departments. Currently there is a limitation to only English names and you cannot delete existing categories.

Position Documents:
These documents can be anything. Just make sure you add only PDF documents. There is one special document which you have to name General Information Letter, this document will be used as intended in the phone book section.
Position Letters Download:

For every position there is the option to generate letters concerning the position. They come empty, and assume to be printed on paper with an ETH-letterhead. Please make sure you choose the correct one – German or English – when you print them.

The *reference tag* refers to the text bit just below the date. The *interview date* refers to that date which is referred to in the body of the letter.

The letter texts can be viewed beforehand and also edited in the *Translation-Section* of the software (near the bottom).

Translations:

Almost every text bit can be translated within the application. The appropriate link for that can be found within *further administrative actions*.

In order to translate stuff to German, one has to first find that bit in the *Original*-column. (Use CTRL-F to quickly find text fragments.) Then click on *create* in order to create a new translation. If you want to change an existing text bit in English, you'll have to create an *English translation* of the original text. That can be done in the same place.

Available languages are German and English. The latter is also the default language. In order to change these properties, the code needs to be changed (see the developer guide).

Updates:

The application supports updating itself from an external repository. Additionally there are more modules available than just the ethire one. Once set up you probably won't use these functions.
8.3 *Developer Guide*

This appendix contains a copy of the developer guide.

**Front-end:**

**Actions:**

The action which is executed depending on the URI has a real counterpart. Method names have a `action_` prepended to the action "action", so for example if the action is *test*, the method `action_test()` will be executed.

All minus-signs in action names are replaced with underscores. (Method names in PHP can not contain minus signs, but they look pretty in URIs).

If the action does not exist (someone playing with the URIs for example), the *not-found* module will be invoked (which displays an error and invokes the default action). The module may redefine this module (`action_not_found()`) in order to do... whatever it wants to do.

**Back-end Connection:**

The database connection for the Ethire-module is built exactly the same as the regular database connection (see below). However, it has an additional database module (inherited from the default one) in order to connect to the Back-end. Since PHP does not support multiple inheritance, it’s implementation is a bit cumbersome: The common code for interfacing with the Eiffel Back-end is in a separate class, but all the database classes have only a reference to it.

Switching between Back-ends can be done from within the application, under further administrative actions. Basically there are some settings which tell the module what classes need to be instantiated.

**Bootstrapping:**

Looking on the root directory, there are only two files. The first one, `index.php`, is responsible for loading the configuration data, the second one (`config.php`) is exactly that.

The main code resides in the module `kernel`. Again, this is configurable. Inside the kernel-module, the file `index.php` is executed, so most of the bootstrapping happens there. Basically, it does the URI stuff, loads classes, executes it and finally loads the `view` and renders the content.

**Data Storage:**

All files for any module is stored in its own directory. The module is responsible for creating any directories which it needs. Also it is responsible for deleting everything once it is not used any more!

The `resource`-module uses the data storage for storing the templates (ETH corporate design templates). They are each in its own directory.
Database Connection:
The main database connection file is located in the kernel (file db.php, class intra_db). Any module may inherit from this class in order to have custom db-connection-methods. It also contains default methods for retrieving a row by id, for retrieving all rows or for inserting, editing and deleting rows.

Ethire module:
Everything related to Ethire is located in this module. Even though it outsources much to the Back-end, it is still quite large. Basically everything is located within its action, be it "candidate", "position", or whatever.

Files:
All code and essential images are in the directory application. Data produced by the software lie in the directory data. Where these directories go depend on the configuration. They can also point to a different computer (via nfs or similar). Within, there is a directory for each module. Inside each module, the files are named like this:

- <module_name>.php: Here goes the class which holds most of the application logic
- <module_name>_db.php: Here goes the class which holds the database interface/implementation
- <module_name>_icon.png: The icon of the module

Of these 3 files, only the first is required. If the database file is missing, the default one will be used, likewise with the icon. All other files within the module have to be included by the module itself! (This usually happens within the init()-method or inside the actions as needed.)

The module is responsible for creating the data directory itself (usually within its install()-method). Inside, it can organize the data itself any way it wants.

Modules:
All the functionality is divided into modules. The division is based on what they are responsible for; the names chosen should indicate what it is. For example, the "resource"-module is responsible for displaying all resources (images, css,...) as well as rendering the xhtml page. The "login"-module is responsible for authentication of the users, and it exposes its functionality to other modules.

If the module does not exist (someone playing with the URIs for example), the default module will be invoked and an error is displayed.

PDF Documents and Templates:
All PDF Documents assume to be printed on paper with the ETH-letterhead already on them. However, the application can still use templates. Currently there is only one template that is used (application/ethire/eth_template_e.pdf). You'll need to enable the use of this template in the class my_pdf, in the Header()-method.
Settings:
The framework provides an easy way to load and save settings for any given module. The settings are loaded and stored in the settings-variable during setup. Storing settings can be done via the database connection. There are methods provided.

Test Data:
In the Front-end there is a section where you can wipe the database and create some test data. Besides testing (or setting up your development system) you'll never need this section. In the code, it is invoked in the method action_import() (ethire-module), the implementation is in the file ethire_import.php.

Translations:
The framework provides a basic translation framework. In order to get a string to be translated, it has to be enclosed in the _() method, which behaves very much like printf() from the C-world. (with the difference of course that the string gets replaced by its translation beforehand.)

The available languages are stored as a property (languages, in the class module near the bottom) in the module. Every language is in one entry of this array. The default language is defined just below.

URI:
The URI defines what is invoked:

/staging/ethire/candidate/3

It can be read like command line parameters of a program. All elements are split by slash (tokenized). Thus, we have

• staging is the base, this always stays the same and points to the main entry point of the application
• ethire is the name of the module (class) which is invoked (called "module")
• candidate determines the name of the method which is invoked (called "action")
• Everything else goes in an array of arguments. Thus, from 3 follows $this->argv[0] is 3.

Updates:
The updater is located in the modules-module. For checking for updates, it looks for it in a repository (configurable via config-file). If there is one, it can download it, unpack it and replace the module with its newer version. The updater may also update itself via the same way. (Note that if there are changes to the API, this may fail depending on the order of which the updates are applied.)

It has a corresponding module called repository. The repository just holds the stuff, does not do anything fancy.

Back-end:
Model-Classes:
Each model-class (EH_*) corresponds to one database table. Every class inherits from EH, which holds the connection stuff and some basic features (for example retrieving exactly one row by its primary key). Some of these model-classes only contain the fields of its database table and nothing more, since more is not needed (yet).

ODBC Connection:
The ODBC connection has to be installed on every computer which houses the Back-end. This has to be done manually on the server itself. See also the installation guide for a more detailed description on how this can be achieved.

Serializer:
The serializer (classes SERIALIZER*) is basically a utility class which has features for converting some basic types to their serialized counterparts. Currently there is just one immediate serializer, but one can add as many as needed.

SQL Connector:
The SQL connector (class SQL_HANDLER) holds everything we need for the database/odbc connection. It's basic feature is the query-feature which takes a well-formatted SQL-query and executes it. After execution, the result is either in result_1 (if only one row is selected, in the case of insertion it's primary key) or result_n (if (potentially) multiple rows are selected).

Database:
Table layout:
The table layout for the databases can be found on the install()-methods in the Front-end. The names of the tables all have a common prefix which is defined in the configuration file. The prefix for Back-end tables can be set within the Front-end, usually it's the same. The prefix can be empty.

The next part of the table name is the name of the module which it belongs to. Then follows either nothing (if the module needs just one table) or any other name the module sees fit. Words are separated by underscores.

As an example, the table xx_ethire_candidate tells you the following: xx_ is the common prefix. ethire is the name of the module, and _candidate the suffix for – most probably something about candidates.
8.4 Installation Guide

This document describes how you can install ethire on a (Debian-Linux) server, using only the terminal.

Apache/PHP/MySQL/...

The lines below assume you want to install all parts of ethire on the server, if not you can safely omit some parts.

```
sudo aptitude update
sudo aptitude safe-upgrade
sudo aptitude install apache2
sudo aptitude install mysql-server odbcinst1
sudo aptitude install php5 php5 php5-imap php5curl php5-mcrypt php5-mysql phpmyadmin
sudo aptitude install subversion
```

EiffelStudio:

This describes installing EiffelStudio for the local user. If there is a new release of EiffelStudio, you have to change the download location accordingly.

```
cd ~
wget ftp://ftp.inf.ethz.ch/pub/mirror/eiffel/63/Eiffel63_gpl_76070-linux-x86.tar.bz2
mkdir Eiffel63
```

```
tar -jxvf Eiffel63_gpl_76070-linux-x86.tar.bz2
export ISE_EIFFEL=~/Eiffel63
export ISE_PLATFORM=linux-x86
export PATH=$PATH:$ISE_EIFFEL/studio/spec/$ISE_PLATFORM/bin
cd Eiffel63
./make_install &
```

```
cd $ISE_EIFFEL/library/store/dbms/rdbms/odbc/Clib
finish_freezing -library
```

Set up Apache:

Go to /etc/apache2/. We need to at least enable following modules:

- cgi
- rewrite
All webserver documents (usually) go to /var/www/.

Frontend:
The documents go to /var/www/front. So,
```
cd /var/www
sudo mkdir front
sudo chmod 777 front
svn co https://svn.origo.ethz.ch/ethire/Source/ethire_front/ front
cd front
cp .htaccess.sample .htaccess
cp config.php.sample config.php
```
Please make sure you edit those two files (.htaccess and config.php) to match your installation.

Back-end:
We compile for the local user, then copy the binary goes to /var/www/cgi-bin. So,
```
cd ~
svn co https://svn.origo.ethz.ch/ethire/Source/ethire_back/ back
cd back
ec -finalize -config ethire_back.ecf
cd EIFGENs/ethire_back/F_code/
finish_freezing
cp ethire_back /var/www/cgi-bin
```
Note there is a small script which does most of the things for you. It's called compile and is located in the top directory of the Back-end.

ODBC:
Configuration info goes to /etc/odbc.ini. Use
```
isql -v DSN_NAME db_username db_password
```
to test the connection. If that works, but the Back-end does not, try compiling and running the esql example in EiffelStudio. It is located in ~/Eiffel63/examples/store/esql.