b
UNIVERSITÄT
BERN

Teamizer

Developing a suitable schedule and poll platform for the sports club Unihockey Lohn

Bachelor Thesis

Andreas Hohler
from
Lohn-Ammannsegg SO, Switzerland

Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

December 2015

Prof. Dr. Oscar Nierstrasz

Software Composition Group
Institut für Informatik und angewandte Mathematik
University of Bern, Switzerland

Abstract

It has come to our attention that the best known scheduling platform Doodle[1] doesn't support scheduling in closed teams the way we would like it. Did you ever get annoyed by the fact, that some invited people never fill in a Doodle schedule or the proposed options don't fit their schedule? In this work we show how such a web application for scheduling in teams could be designed to meet our requirements.

We describe a software project for a specific sports club called *Unihockey Lohn*[2] which is interested in making scheduling easier for the members and the responsible.

Our web application provides a simple platform for making polls and scheduling. It's based on closed teams with members, so people get notified when a new poll is created or a deadline is near.

[1] `http://doodle.com`
[2] `http://unihockeylohn.ch`

Acknowledgements

My special thanks to *Prof. Dr. Oscar Nierstrasz* for giving me the opportunity to do this software project as a bachelor thesis in the Software Composition Group[3] of the University of Berne.

I would like to thank my sports club *Unihockey Lohn* for making this bachelor thesis possible and give me a chance to revolutionize the scheduling in this club.

Finally, I appreciate the help of *Karan Sethi* and *Linus Schwab* for testing my work and making good suggestions to improve the system.

[3] `http://scg.unibe.ch`

Contents

1 Introduction

Today, efficient and easy scheduling is important to run an organization even if it's quite small. There exist many tools that have the intention to help manage this difficulty.

The main idea of Teamizeris to get better response on polls by explicitly addressing all target persons. To achieve this, one creates a group and adds existing accounts or imports new users by mass processing email addresses and names. In a group people can be divided into sub-groups with the help of ranks. Thus distinct polls are supported.

Furthermore a notification system will keep the user informed about activities in groups and polls. For example if a new poll is created or the poll was closed, the user receives an email and a web notification. And to reduce the difficulty of learning a new system, a poll has similar features like Doodle already provides.

We faced many challenges particularly when trying to implement non-trivial procedures in Symfony because the framework has useful but also obstructive constraints a programmer must take into account.

In conclusion, the project was successful. We could implement all requirements and got a working application that is ready to use. But one big requirement had to be put aside in order to not excessively exceed the time plan: the mobile application support. This is a future extension of the project we will seriously consider.

We improved our knowledge about how to face and deal with technical and software engineering challenges. We learned that sometimes requirements should be prioritized in order to finish a project in the foreseeable future. Furthermore we learned how a complex PHP framework works and the way a Model-View-Controller designed application should look like.

2
Related Work

Doodle

Doodle[4] is the best known scheduling platform in Switzerland and has its origin also in Switzerland. Worldwide over 20 million users per month use the service with one Doodle being created every two seconds.

It can be used quickly and easily to find a date and time to meet with multiple people. First, you suggest dates and times the participants can choose from. Every participant selects dates and times from the poll that he is free and Doodle will compose the responses and tell which option works best for everyone. (Source: Doodle Blog[5])

But there are also some features we miss and would like to have. It does not really provide a possibility to continuously create polls in a group of people, where they have a good overview over group polls and deadlines.

Teamplanbuch

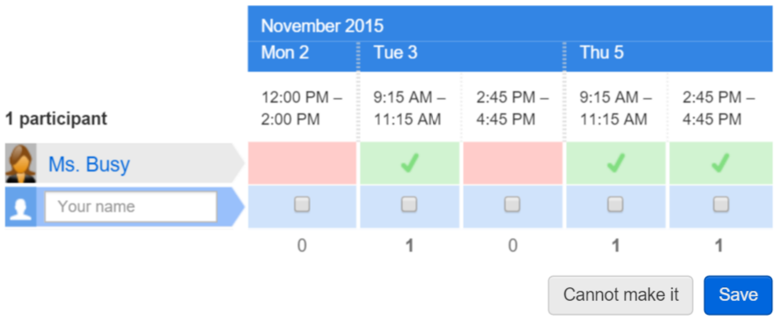
The Teamplanbuch[6] is a Swiss website that helps organize clubs and teams. It enables people to register or deregister from events, trainings, meetings or plays. It offers great

[4] `https://doodle.com`

[5] `http://en.blog.doodle.com/2014/01/29/doodle-crosses-the-20m-user-mark/` 21.09.2015

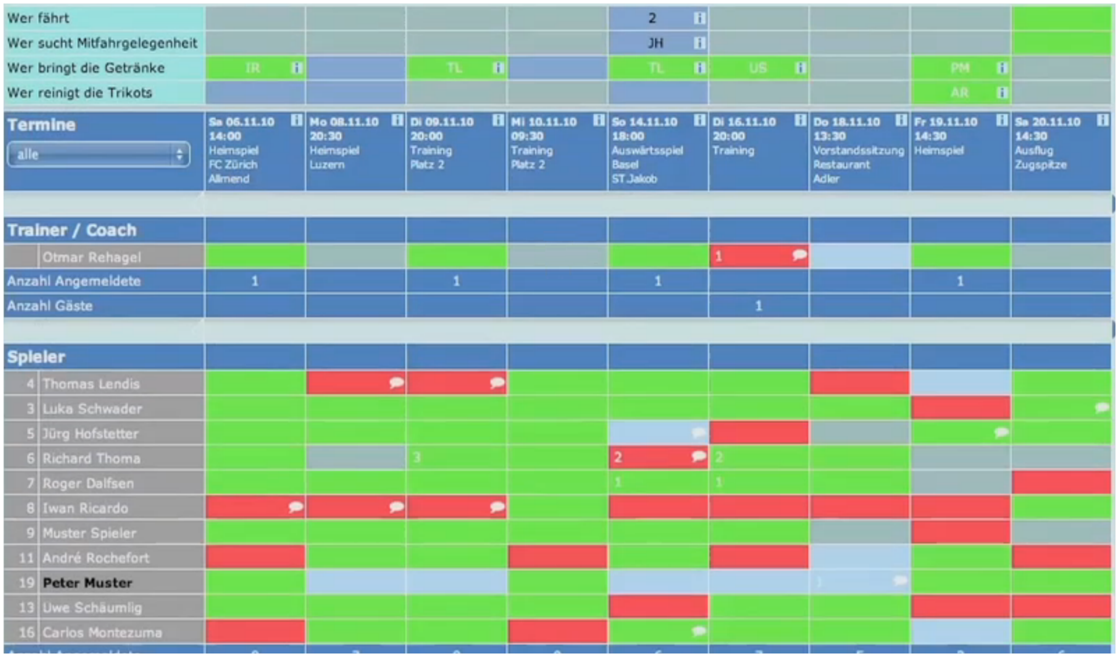
[6] `https://www.teamplanbuch.ch`

features but also lacks some requirements like sub teams, the simplicity of Doodle and the creation of simple polls.



	November 2015				
	Mon 2	Tue 3		Thu 5	
1 participant	12:00 PM – 2:00 PM	9:15 AM – 11:15 AM	2:45 PM – 4:45 PM	9:15 AM – 11:15 AM	2:45 PM – 4:45 PM
Ms. Busy		✓		✓	✓
Your name	☐	☐	☐	☐	☐
	0	1	0	1	1

Figure 2.1: Doodle example



Wer fährt					2				
Wer sucht Mitfahrgelegenheit					JH				
Wer bringt die Getränke	IR		TL		TL	US		PM	
Wer reinigt die Trikots								AR	
Termine alle	Sa 06.11.10 14:00 Heimspiel FC Zürich Allmend	Mo 08.11.10 20:30 Heimspiel Luzern	Di 09.11.10 20:00 Training Platz 2	Mi 10.11.10 09:30 Training Platz 2	So 14.11.10 18:00 Auswärtsspiel Basel ST.Jakob	Di 16.11.10 20:00 Training	Do 18.11.10 13:30 Vorstandssitzung Restaurant Adler	Fr 19.11.10 14:30 Heimspiel	Sa 20.11.10 14:30 Ausflug Zugspitze
Trainer / Coach									
Otmar Rehagel	🟩		🟩		🟩	🟥 1		🟩	
Anzahl Angemeldete	1		1		1			1	
Anzahl Gäste						1			
Spieler									
4 Thomas Lendis	🟩	🟥	🟥	🟩	🟩	🟩	🟥		🟩
3 Luka Schwader	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟥	🟩
5 Jürg Hofstetter	🟩	🟩	🟩	🟩		🟥		🟩	🟩
6 Richard Thoma	🟩		🟩 3	🟩	🟥 2	🟩 2	🟩		
7 Roger Dalfsen	🟩	🟩	🟩	🟩	🟩 1	🟩 1	🟩		🟥
8 Iwan Ricardo	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟩
9 Muster Spieler	🟩	🟩	🟩	🟩	🟩	🟩		🟥	
11 André Rochefort	🟥	🟩	🟩	🟥	🟩	🟥		🟩	🟥
19 **Peter Muster**	🟩			🟩				🟩	🟩
13 Uwe Schäumlig	🟩	🟩	🟩	🟩	🟥	🟩	🟩	🟥	🟥
16 Carlos Montezuma	🟥	🟩	🟩	🟥	🟩	🟩	🟩		🟩

Figure 2.2: Teamplanbuch example

Conclusion

After considering both Doodle and Teamplanbuch we are able to collect shortcomings and build essential features with the gathered knowledge.

Both services lack a good notification system. Doodle doesn't really support building groups with members. Teamplanbuch implements this feature but is rather overly complicated and the simplicity of Doodle is missing.
Based on this, we have the following key features our application should cover:

- Notification system for groups and polls
- Creating groups of people to efficiently distribute the poll to invited users
- Text polls and schedule polls
- No overloaded application - a lightweight dashboard where you just see the relevant things. The screen should not be full of different colors like Teamplanbuch.
- Straightforwardness like providing the wanted functions in an understandable way for users and a big part of Doodle's poll features in general

3 The Problem

Based on a meeting with the board members of Unihockey Lohn and experience with currently used scheduling solutions, we discovered the following functional and non-functional requirements.

3.1 Functional requirements

These requirements specify the functionality of the system.

Authentication & Authorization

Like every user-specific website, a good authentication and authorization system is needed. To enable later extensions of an administration area, it should support different roles like administrators and users.

It must implement some default requirements like password recovery, user profile and of course account creation.

Our research led to the knowledge that Symfony already contains an authentication service that supports roles and just needs to be modified for working with our own user entity. This was an important requirement but easy to implement.

Polls

The main project goal was the implementation of creating text and scheduling polls. In a text poll one can simply enter text options a participant will be able to choose from. In a schedule poll date options can be set with the help of a date picker. For each date option there should be a field where a specific time or text can be added.

Participants should be able to add comments to the poll and edit their entries as long as the poll is open.

The poll initiator can choose the final options that “won” the poll and will therefore close the poll if it was not already closed by the deadline.
The poll creation process should look like the following:

1. Title, Description, Deadline, Location
2. Poll options which participants can choose
3. Additional settings
    - Participants are allowed to only choose one option
    - Entries are hidden, no one can see the answers of others
    - Maximum of participants that can choose a specific option
    - Possibility to make “if-need-be” available for busy people (This means that the participant can mark a not perfect option as if-need-be, and not as “can’t”)
4. Choose the target sub-groups of a group if there are any
    - The members of a specific ranks can view or/and participate in the poll

The poll participation process and back end logic were challenging to implement due to Symfony’s form construction and MVC restrictions. On the other hand the comment part had a low priority and its implementation was easy.

Groups

Another non-negligible requirement was the group system. Groups allow us to specifically invite a bunch of people to polls and notify them about changes. The challenging part of implementing this feature was to get the adding members form working. We again had to get along with Symfony’s form system.

The sub-group functionality was provided with a rank system, where users can get assigned to specific ranks. A rank is a sub-group of the original group where users are part of. Polls can be created with specific ranks as targets.

Notification

To ensure that users are up to date with changes and new polls, we need a notification system. This was also declared as a key feature, especially for further development of mobile applications. We decided later to exclude the mobile applications from the thesis. Notifications must be visible on the web application and should be delivered via email. This requirement was implemented last and brought some difficulties.

Calendar

The user should have a nice calendar where all deadline and fixed dates are shown. Additionally an iCal feed should be provided to which clients can subscribe. This was not a risky requirement because it was low prioritized.

3.2 Non-functional requirements

These requirements specify the quality of the system and are mostly related to the satisfiability of the user.

Responsive design

The website should be designed mobile first so the template must be responsive to different resolutions. There are many users who don't use a computer to access the internet but instead use their mobile phone.

Technology

The application is written in PHP so there isn't really a special environment needed to get the application working. Because of this we restricted ourselves to only evaluate PHP frameworks.

Database

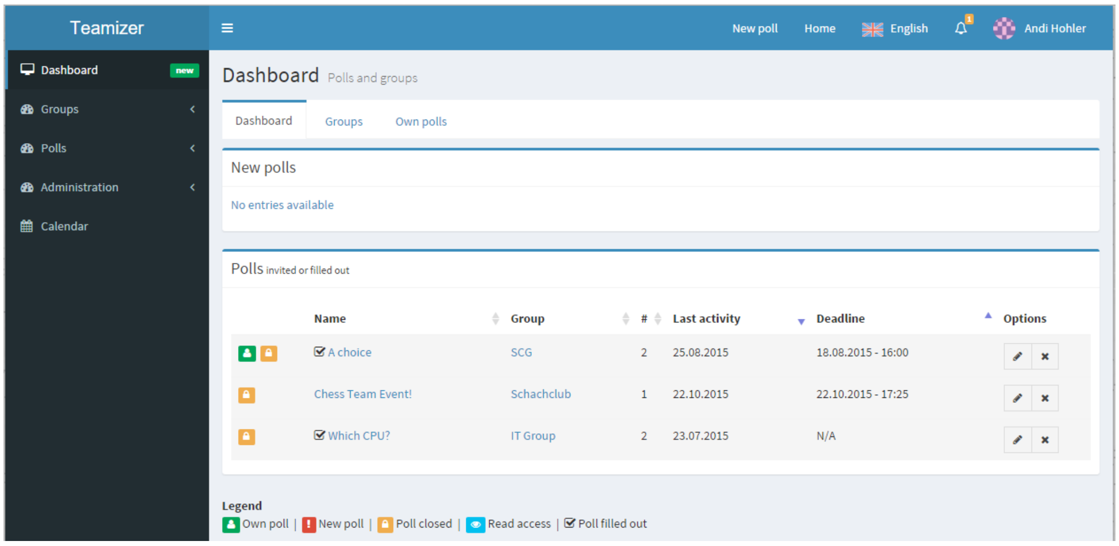
The site response time should be adequately low if there are many database requests. The used ORM (Object-relational mapping) system by Symfony doesn't fully load objects from the database. Instead it just gets the currently used fields. This results in many database queries. For example, the dashboard creates 100 queries needing totally 117ms to execute on our local test environment.

The impact on our production environment is not tangible for the user, because it comes with a lot more power than the test environment.

4
The Solution

4.1 User experience

In this section we show some GUI elements and how important functions are implemented.



New polls

No entries available

Polls invited or filled out

	Name	Group	#	Last activity	Deadline	Options
	✓ A choice	SCG	2	25.08.2015	18.08.2015 - 16:00	
	Chess Team Event!	Schachclub	1	22.10.2015	22.10.2015 - 17:25	
	✓ Which CPU?	IT Group	2	23.07.2015	N/A	

Legend
Own poll | New poll | Poll closed | Read access | ✓ Poll filled out

Figure 4.1: Page overview

4.1.1 Authorization

Registration

The registration form consists of the elements “First name”, “Last name”, “E-Mail” and “Password”. A user account has to be unique based on its email address because the address is used in the login process.
To prevent bots from creating accounts we use the new Google reCAPTCHA[7] called *No CAPTCHA reCAPTCHA*. With this, most users can attest they are human without having to solve a real Captcha. Instead they confirm they are not a robot with just a single click. If the Captcha doubts the user’s authenticity, he has to answer a question. For example “Click on all images showing a car”.
With this advanced captcha the user won’t have to confirm his account through a link sent to his Email address.

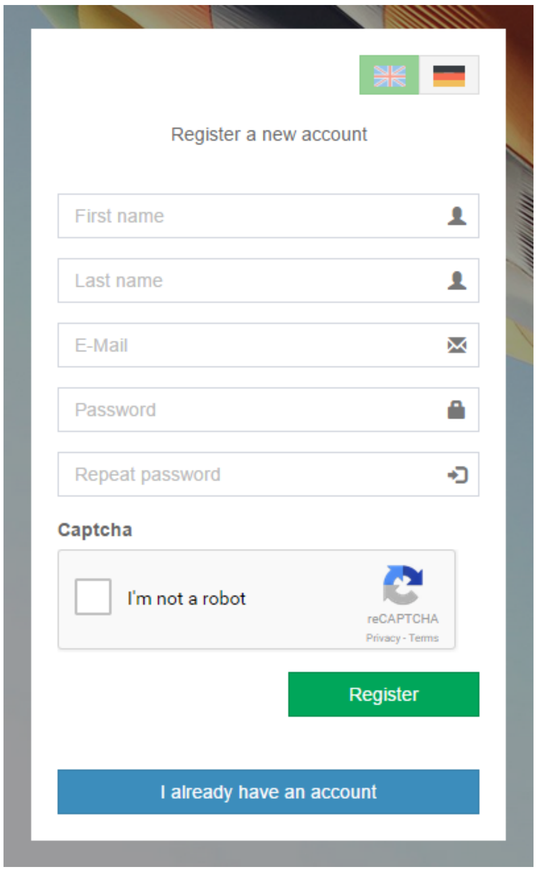


Figure 4.2: Registration

[7]`https://www.google.com/recaptcha`

4.1.1.1 Login

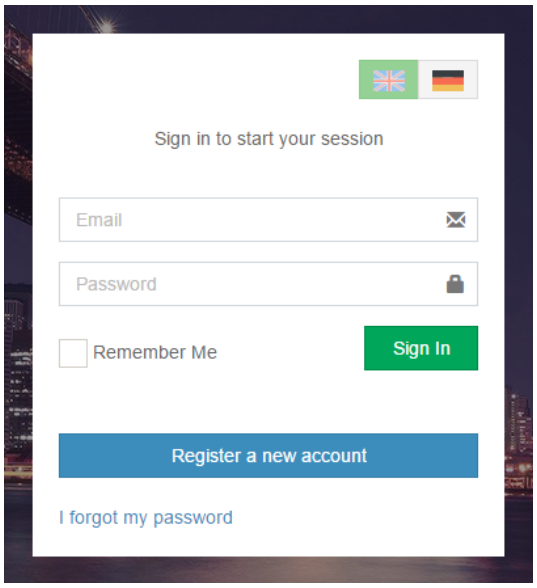


Figure 4.3: Login

The login form is really simple and self-explaining. It is possible to check the "Remember Me" box. So the user stays logged in until the cookies in his browser get removed or he logs off and terminates his session.

Also, a password recovery form is available: the email address and the Google reCAPTCHA is required to start a password restore. Then the user receives a confirmation link over email. After opening it, a new password is generated and delivered again over email.

4.1.2 Dashboard

4.1.3 Groups

4.1.3.1 Create & edit a group

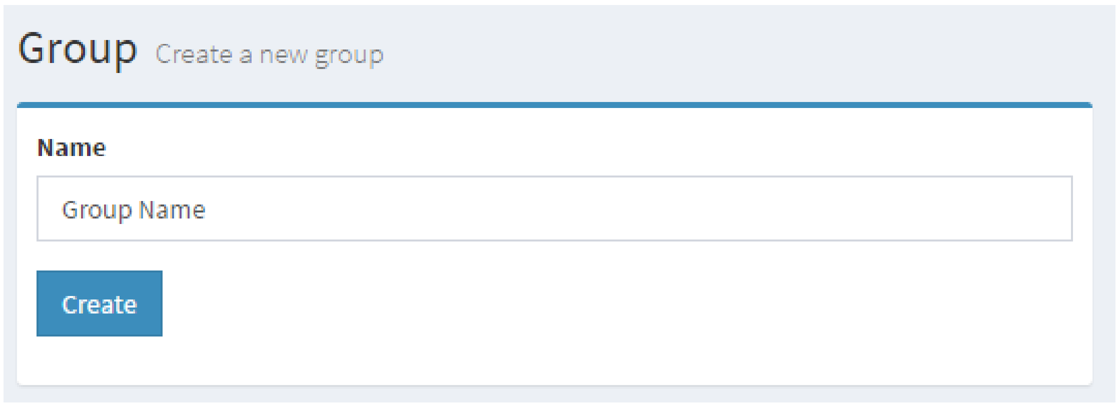
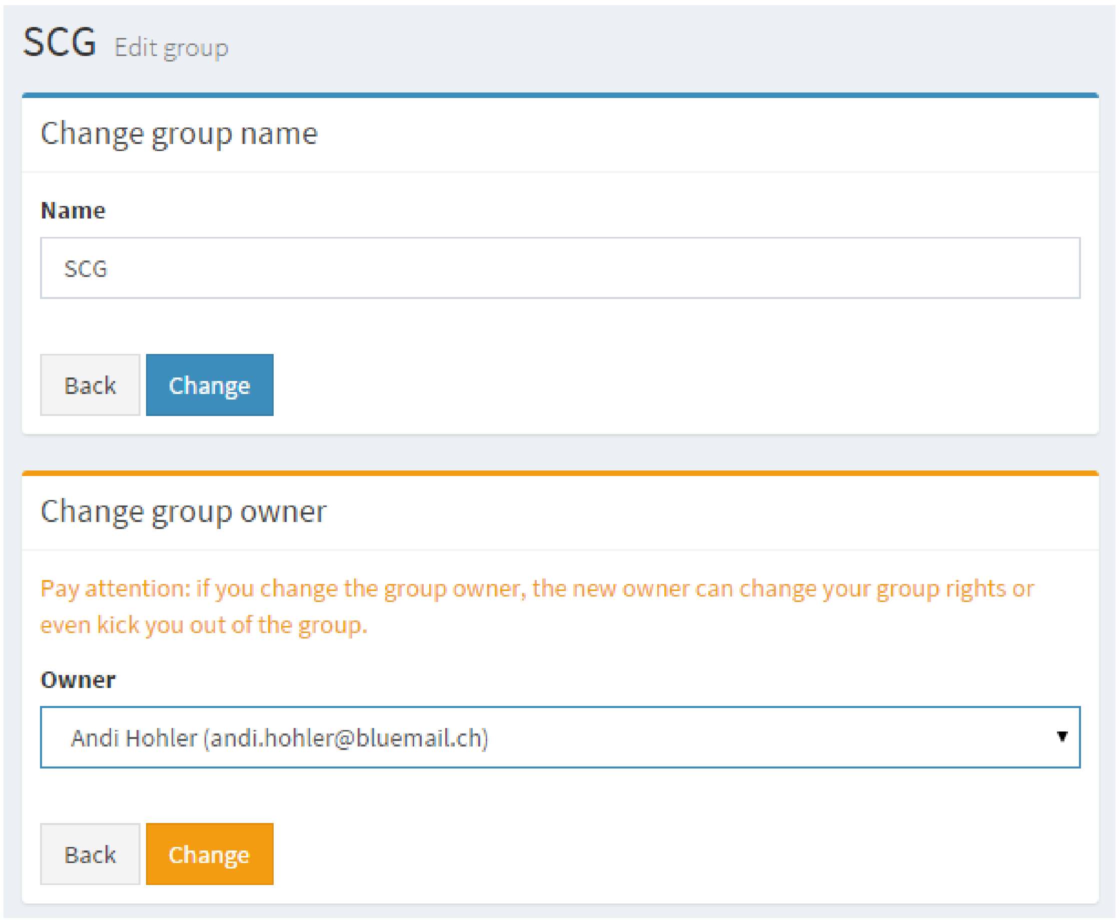


Figure 4.4: Create a group

The group creation process (Figure 4.4) is very basic, we just have to type in a name.

The name doesn't have to be unique, since we don't want to restrict the user.

The next two images in Figure 4.5 show the group editing possibilities. Of course the name can be changed. Also switching the group owner to another group member is possible.


SCG Edit group

Change group name

Name: SCG

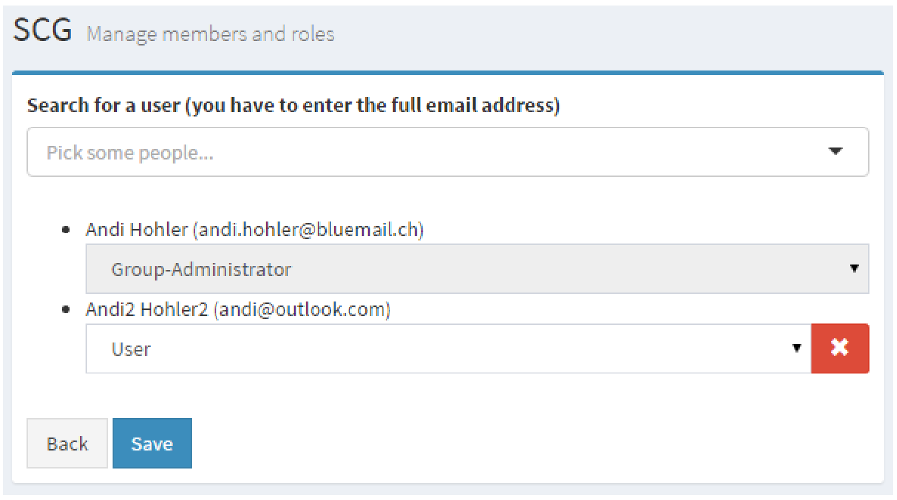
Change group owner

Pay attention: if you change the group owner, the new owner can change your group rights or even kick you out of the group.

Owner: Andi Hohler (andi.hohler@bluemail.ch)

Figure 4.5: Edit a group

4.1.3.2 Members


SCG Manage members and roles

Search for a user (you have to enter the full email address)

Pick some people...

- Andi Hohler (andi.hohler@bluemail.ch)
  Group-Administrator
- Andi2 Hohler2 (andi@outlook.com)
  User

Figure 4.6: Members

To add existing users to a specific group only the corresponding email is necessary. For security purposes the form field does not propose users until the whole email address is entered. The auto-complete field is implemented with select2[8]. This was a bit tricky to implement because we load new users dynamically into the form content and then additionally have to insert the role dropdown field. The challenge continued in the controller logic where we had to combine new and existing members and assign the roles to them. Within this form the roles of members can be changed. Possible roles are “User”, “Moderator” and “Group-Administrator”. The Moderator is allowed to create polls and manage them. The Group-Administrator has full access to all polls in the group.

We also have the possibility to add several users concurrently with the mass form. If a user doesn’t exist (only email address is verified) the account is automatically created and an email with the credentials is sent.

4.1.3.3 Ranks

With a rank system we are able to control access to polls inside the group. For example our sports club has 3 teams. But sometimes we just want to create a poll for one or two of the teams. This system makes it really easy to handle such needs. After creating some ranks we can assign them to group members. Ranks are seen as sub-groups of a group.

[8] `https://select2.github.io` 07.03.2015

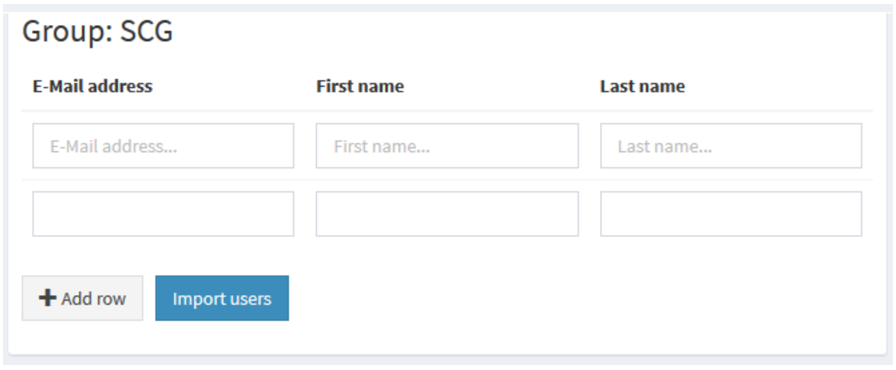


Figure 4.7: Import users

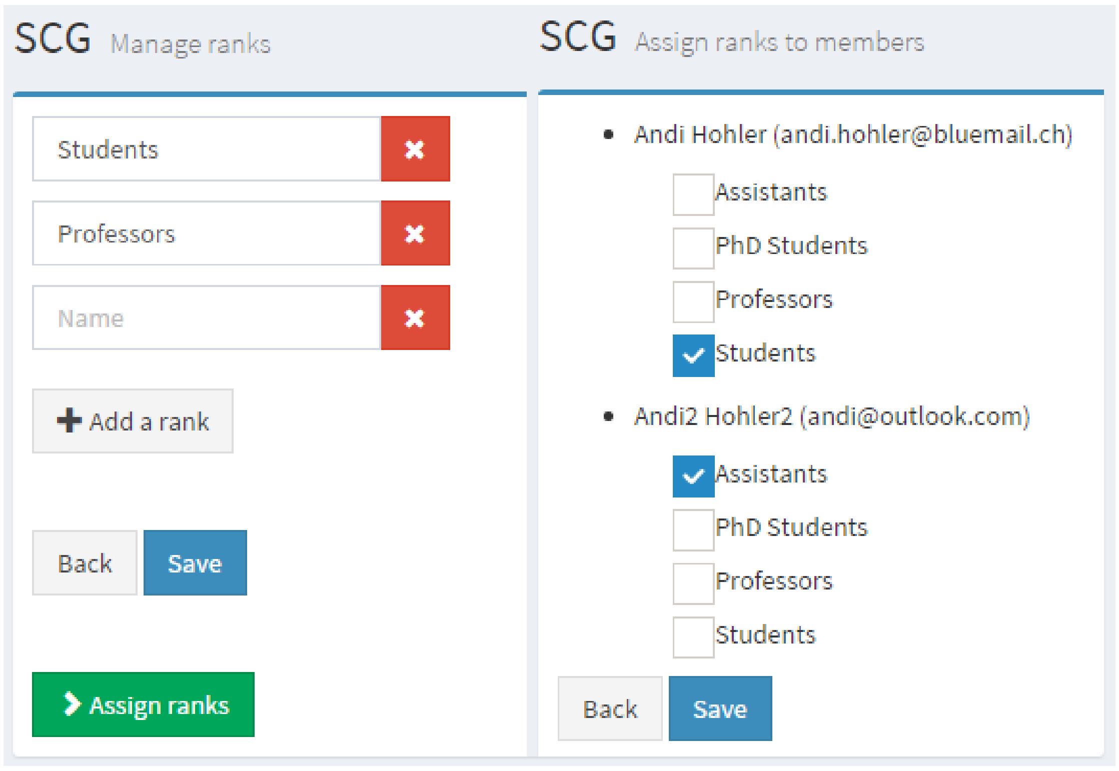
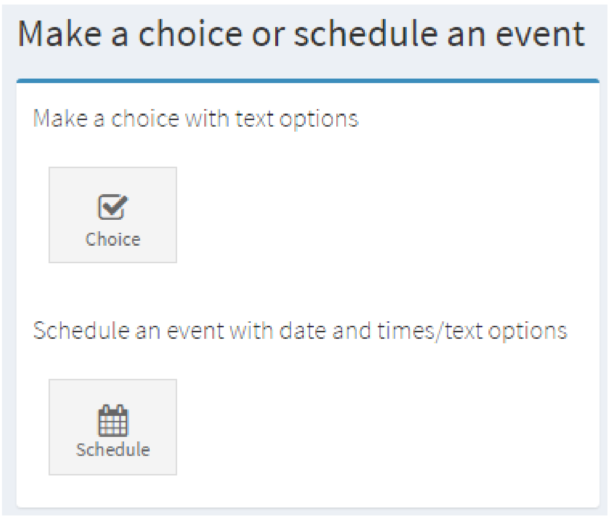


Figure 4.8: Ranks

4.1.4 Polls

4.1.4.1 Create a poll

Poll type
This is the entry point of creating a new poll. The user can choose between text and schedule polls. In a text poll (or choice) the user can simply add text options, whereby in a schedule poll an extended date picker with time is implemented.


Make a choice or schedule an event

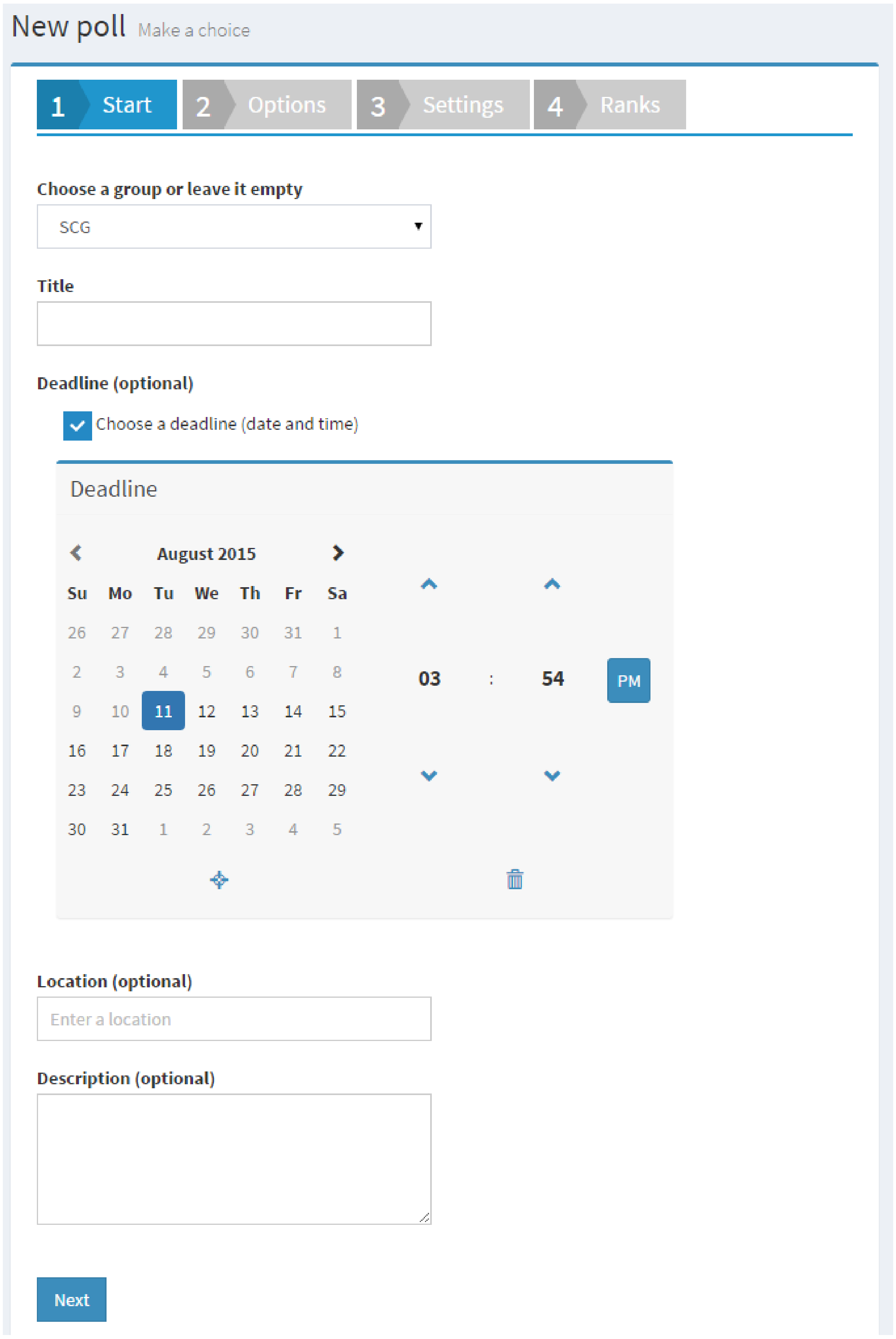
Make a choice with text options

Schedule an event with date and times/text options

Figure 4.9: Choice or schedule poll

Poll wizard
The following image shows the first step (Figure 4.10) of the poll creation process. We used a Symfony bundle to implement the wizard at the beginning, but it was rather complicated and not very stable. Also, every step required reloading the site. So we switched to another solution with the Ideabox[9] JavaScript wizard. It already has a nice design and easy step flow. It supports a responsive view and has variable color schemes. We choose the blue theme because it goes well with the site template. Additionally to make the checkboxes fancier, we included the iCheck[10] plugin.
The deadline picker is implemented with the Bootstrap 3 Datepicker[11] that also supports time. The location field is auto-complete and linked with the Google Maps Places library. We used a plugin[12] that mixes jQuery Autocomplete and Google Maps Places API. It should be noted that we have to configure a seperate Google API Key on every host where we deploy the application. To prevent the user from losing unsaved data, if he tries to close the window, he is prompted to confirm it first.

[9]`http://codecanyon.net/item/ideabox-multipurpose-step-form/10818199` 16.05.2015
[10]`https://github.com/fronteed/iCheck` 16.05.2015
[11]`https://eonasdan.github.io/bootstrap-datetimepicker/` 31.03.2015
[12]`http://www.jqueryscript.net/form/jQuery-Location-Autocomplete-with-Google-Maps-Places-Library-Placepicker.html` 31.03.2015



New poll Make a choice

1 Start 2 Options 3 Settings 4 Ranks

Choose a group or leave it empty

SCG

Title

Deadline (optional)

✓ Choose a deadline (date and time)

Deadline

August 2015

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

03 : 54 PM

Location (optional)

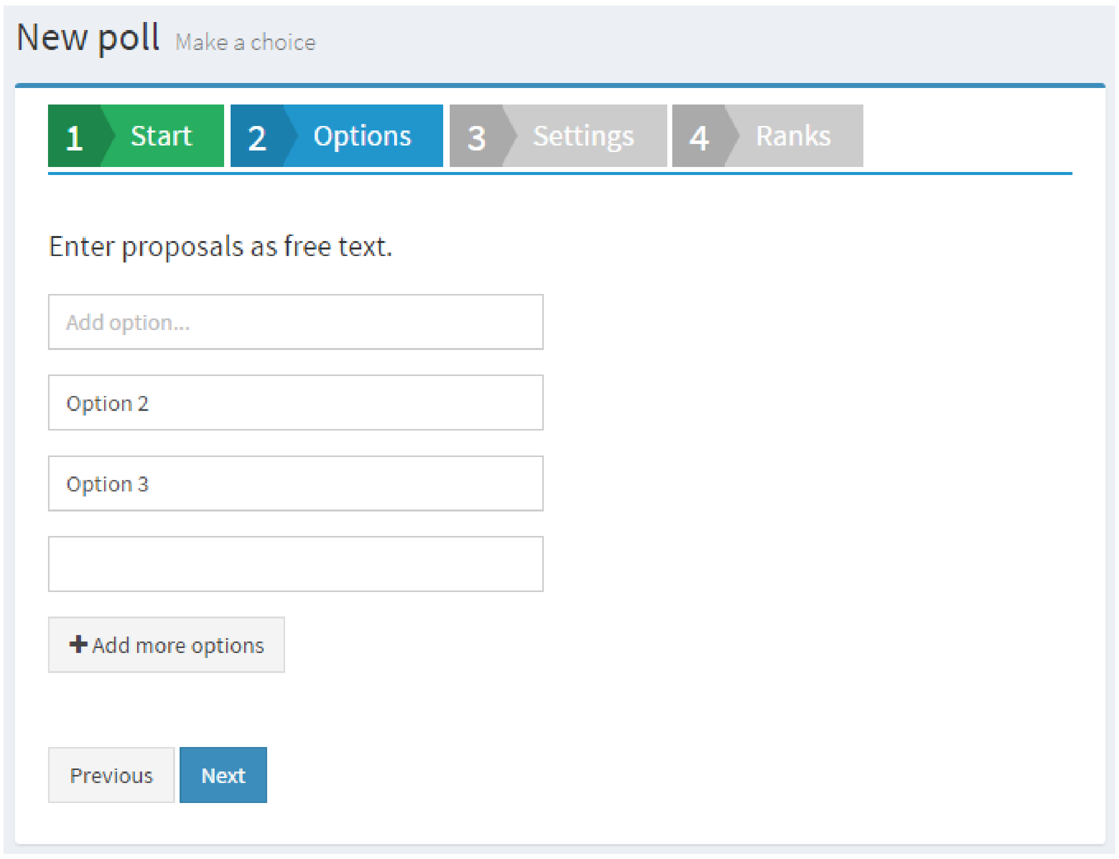
Enter a location

Description (optional)

Next

Figure 4.10: Create a poll

Set poll choice options



Enter proposals as free text.

Add option...

Option 2

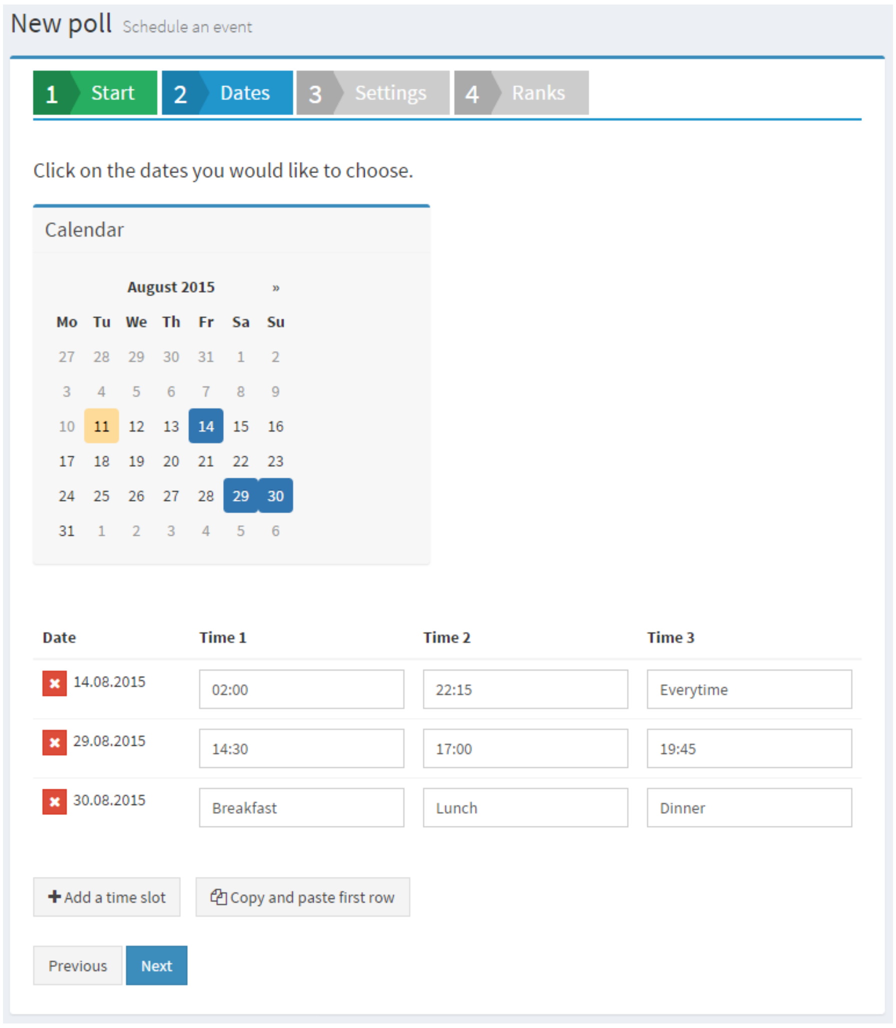
Option 3

Figure 4.11: Set poll choice options

The second wizard step (Figure 4.11) shows either input fields to enter proposals, or a date picker with additional time/text fields. The choice poll is very simple: one can add more fields if needed. If not all fields are used, they are simply left empty.
To add more input fields, the Symfony form collection component[13] provides a prototype function that gives us a form skeleton, where we only have to insert an incremented number into the input name attribute.

[13]`http://symfony.com/doc/current/cookbook/form/form_collections.html`

Set poll schedule options



Click on the dates you would like to choose.

Date	Time 1	Time 2	Time 3
14.08.2015	02:00	22:15	Everytime
29.08.2015	14:30	17:00	19:45
30.08.2015	Breakfast	Lunch	Dinner

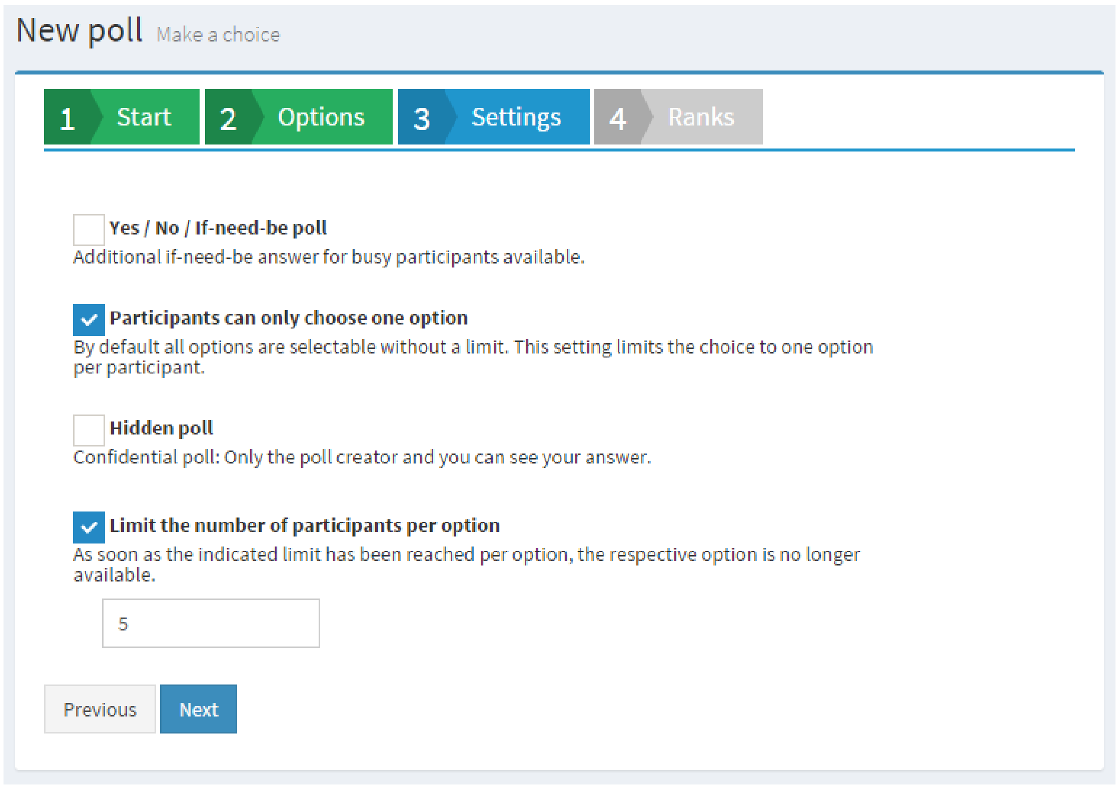
Figure 4.12: Set poll schedule options

For implementing the date picker, we used another plugin named Bootstrap Datepicker[14]. It gives us the possibility to select multiple dates and manipulate the selected data through methods. Additional time fields can be used as text or time input. Times are automatically formatted by Moment.js[15], which allows us to permit different time input formats. This was a bit difficult to implement since a user can insert different time formats. So the input fields must get checked against different formats to parse the time and get valid Moment objects.

[14]`https://github.com/eternicode/bootstrap-datepicker` 15.05.2015
[15]`http://momentjs.com`

Furthermore, functions are provided to copy the first row to all others and to add more time columns. This feature was rather challenging to implement, because we had to find a way to subscribe to datepicker manipulations in order to update the input fields.

Set poll settings



☐ **Yes / No / If-need-be poll**
Additional if-need-be answer for busy participants available.

✓ **Participants can only choose one option**
By default all options are selectable without a limit. This setting limits the choice to one option per participant.

☐ **Hidden poll**
Confidential poll: Only the poll creator and you can see your answer.

✓ **Limit the number of participants per option**
As soon as the indicated limit has been reached per option, the respective option is no longer available.

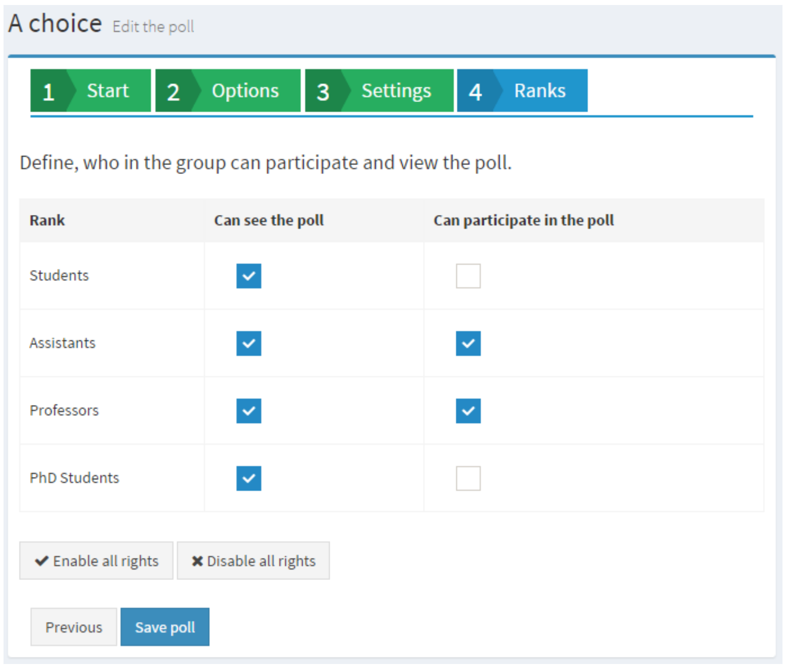
5

Figure 4.13: Set poll settings

The settings are equal for both poll types. We provide some essential settings:

- Yes / No / if-need-be poll: Participants can choose between three options. An if-need-be option is displayed orange and states that the specific option suits the participant if it's absolutely necessary.
- Participants can only choose one option: This setting is disabled, if if-need-be is enabled.
- Hidden poll: The participants can't look at other participant's answers.
- Limit the number of participants per option: This function only works if the if-need-be setting is disabled.

Set poll access rights

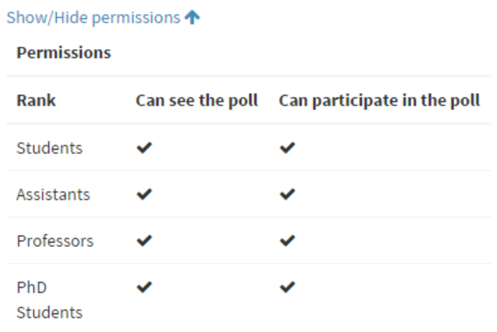


Define, who in the group can participate and view the poll.

Rank	Can see the poll	Can participate in the poll
Students	✓	☐
Assistants	✓	✓
Professors	✓	✓
PhD Students	✓	☐

Figure 4.14: Set poll access rights

If the group owner has added new ranks, the poll creator can define which ranks are allowed to see the poll or to participate in the poll.

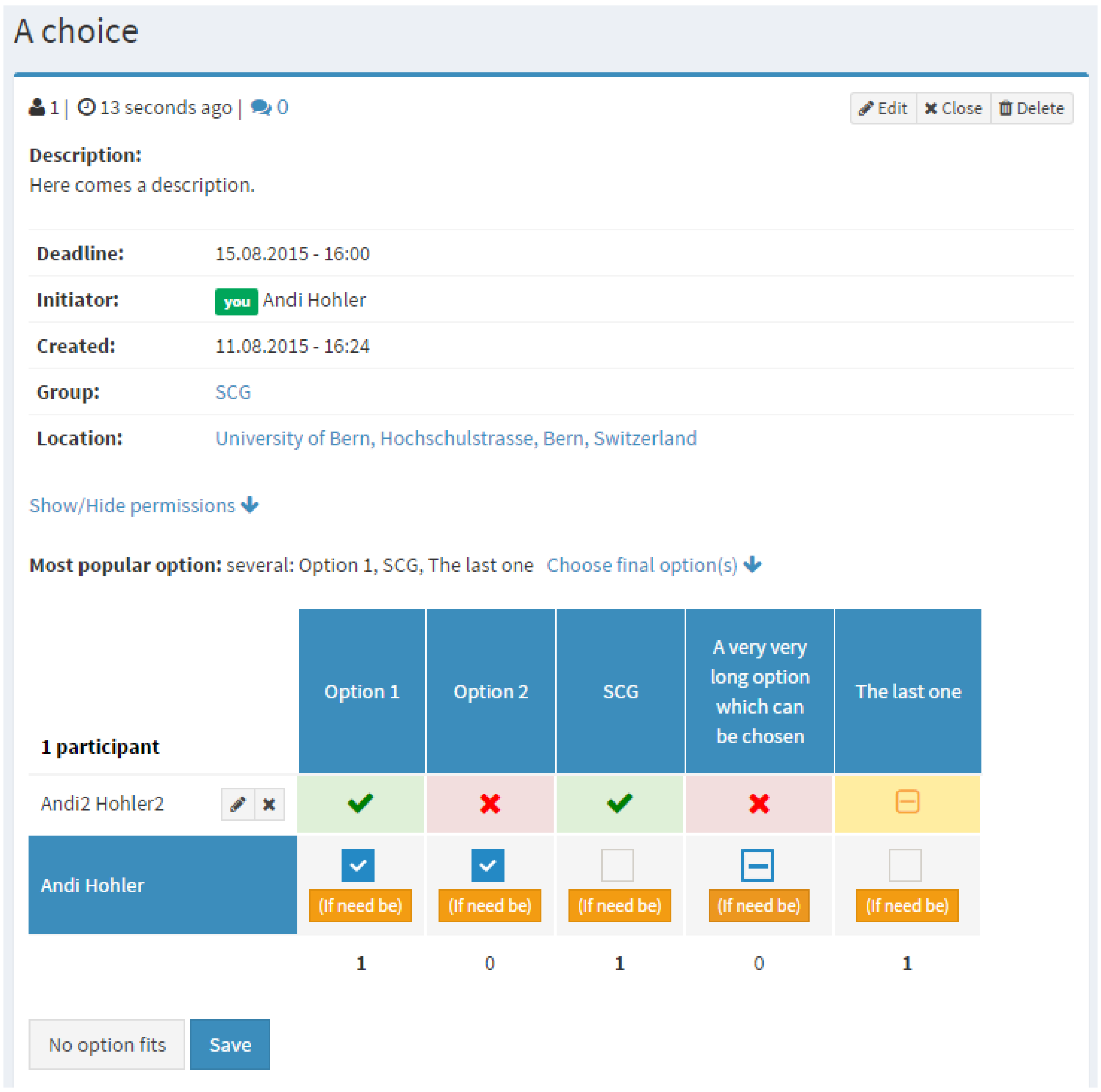


Permissions

Rank	Can see the poll	Can participate in the poll
Students	✓	✓
Assistants	✓	✓
Professors	✓	✓
PhD Students	✓	✓

Figure 4.15: Poll permissions shown in poll view

4.1.4.2 Poll view



A choice

1 | 13 seconds ago | 0

Edit Close Delete

Description:
Here comes a description.

Deadline:	15.08.2015 - 16:00
Initiator:	you Andi Hohler
Created:	11.08.2015 - 16:24
Group:	SCG
Location:	University of Bern, Hochschulstrasse, Bern, Switzerland

Show/Hide permissions

Most popular option: several: Option 1, SCG, The last one Choose final option(s)

1 participant	Option 1	Option 2	SCG	A very very long option which can be chosen	The last one
Andi2 Hohler2	✓	✗	✓	✗	⊟
Andi Hohler	☑ (If need be)	☑ (If need be)	☐ (If need be)	⊟ (If need be)	☐ (If need be)
	1	0	1	0	1

No option fits Save

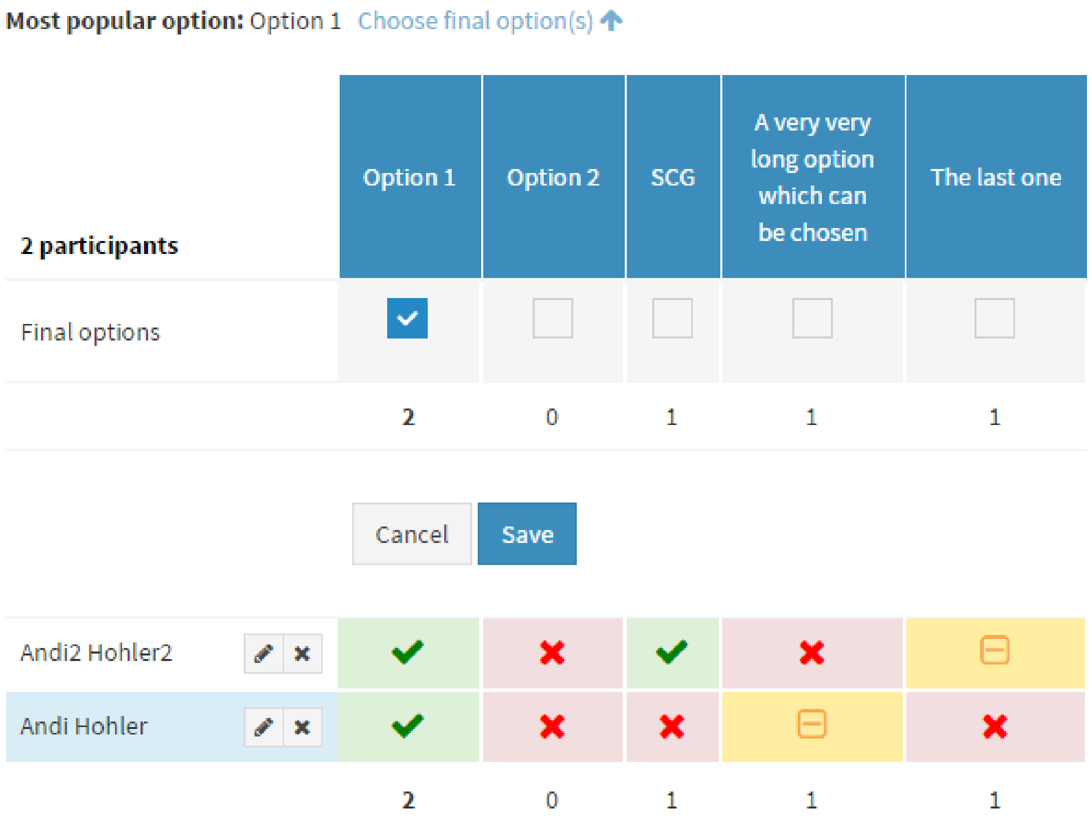
Figure 4.16: Poll view

The header of the poll view includes description, deadline, initiator, group, location and permissions. In the main part, the poll options and participants are displayed. The current user is highlighted with blue color on the left. Entries of users are colored after their states (Yes: green, No: red, if-need-be: orange).

The user can simply use the “No option fits” button if he doesn’t want to participate. The meaning of our polls is, that people fill out the poll, even if no proposal fits their interest.

Here was challenging again the implementation of the normal Yes/No checkbox and the if-need-be button because these are two seperate entity fields where the order of both has to match. The form engine of Symfony would mess up the order if we not directly render the corresponding form field with the Yes/No checkbox entity value as argument.

4.1.4.3 Fix poll options



Most popular option: Option 1 Choose final option(s)

2 participants	Option 1	Option 2	SCG	A very very long option which can be chosen	The last one
Final options	☑	☐	☐	☐	☐
	2	0	1	1	1

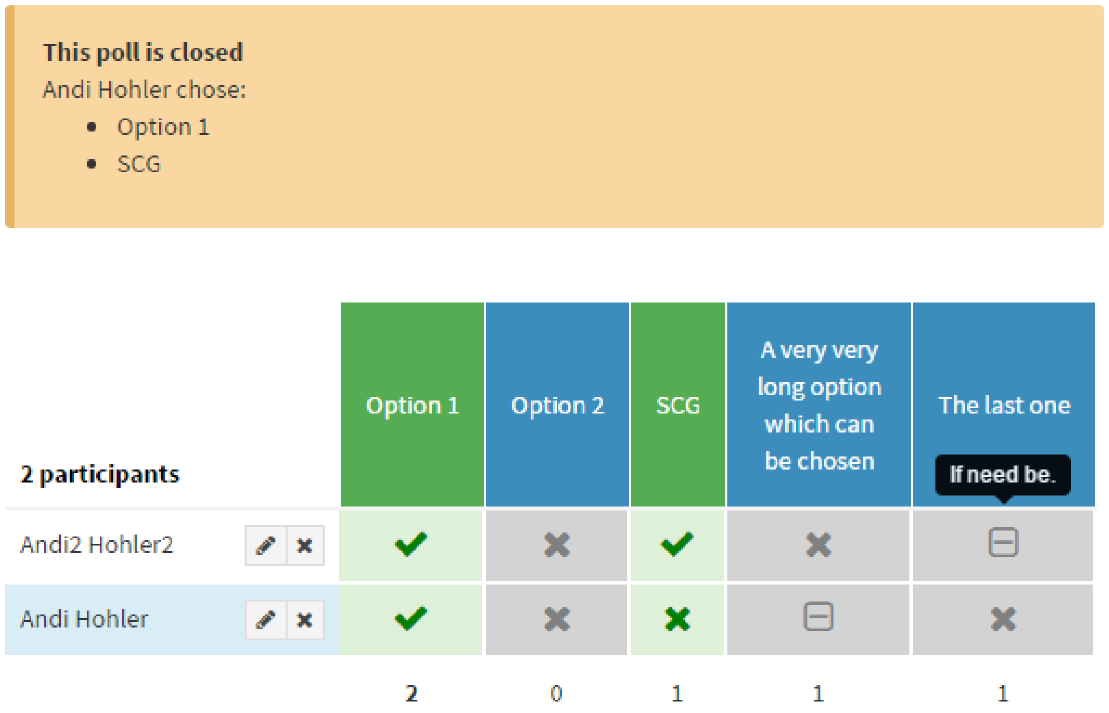
Cancel Save

	Option 1	Option 2	SCG	A very very long option which can be chosen	The last one
Andi2 Hohler2	✓	✗	✓	✗	⊟
Andi Hohler	✓	✗	✗	⊟	✗
	2	0	1	1	1

Figure 4.17: Fix poll options

The poll initiator can close the poll any time he wants to. He can do it with a simple click on "Close" or with choosing the final options. These options will be highlighted in green, so any viewer can see the result. The system proposes the most popular option to simplify the decision.

After a poll is closed, the poll owner can reopen it. If a deadline was set and the deadline has passed, a new one must be set in order to be able to open the poll again. How this looks is shown in Figure 4.18 on the next page.



This poll is closed
Andi Hohler chose:

- Option 1
- SCG

2 participants	Option 1	Option 2	SCG	A very very long option which can be chosen	The last one If need be.
Andi2 Hohler2	✓	✗	✓	✗	⊟
Andi Hohler	✓	✗	✗	⊟	✗
	2	0	1	1	1

Figure 4.18: Fixed options & closed poll

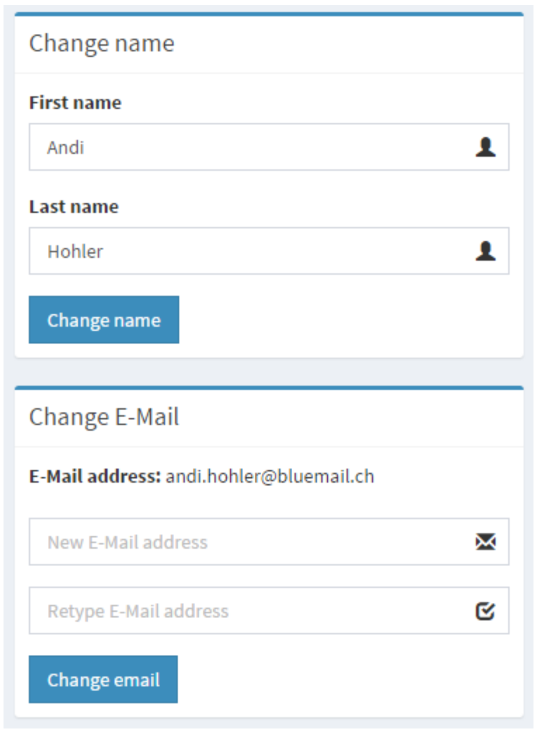
4.1.4.4 User profile

The user profile is kept simple. It allows the user to change details like first & last name, E-Mail address and password (Figure 4.19). To change the password the user must fill in the current one and obviously the new one twice.

A new avatar can be generated (Figure 4.20) and if the user likes it he can save it. We use the DwrAvatarBundle[16] for Symfony to generate and save avatars.

There is also the possiblity to delete the account. The account isn't getting hard deleted. It's just a flag so the user can't login anymore. Because if we would remove the account from the database, some entity relations would be invalid. For example in the poll view: the entry of the deleted user should be displayed for consistency reasons.

[16]`https://github.com/dariuszwrzesien/DwrAvatarBundle` 01.04.2015



Change name

First name

Andi

Last name

Hohler

Change name

Change E-Mail

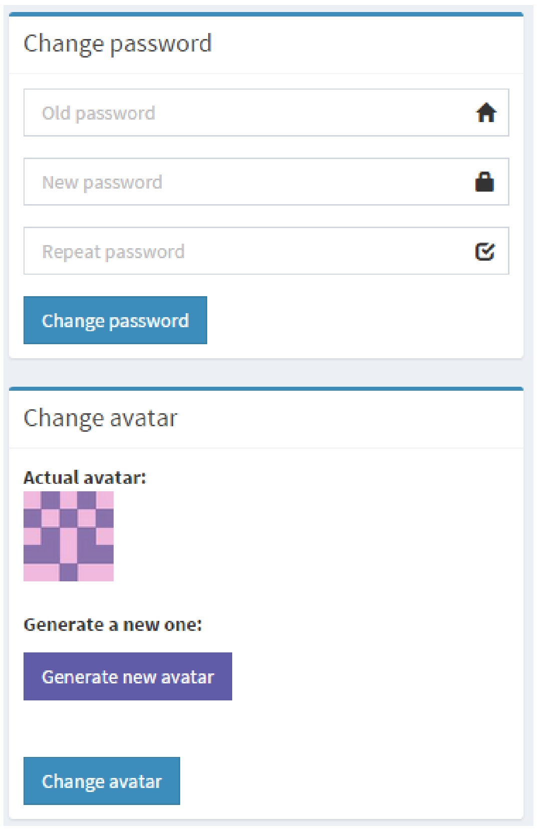
E-Mail address: andi.hohler@bluemail.ch

New E-Mail address

Retype E-Mail address

Change email

Figure 4.19: Edit profile 1



Change password

Old password

New password

Repeat password

Change password

Change avatar

Actual avatar:

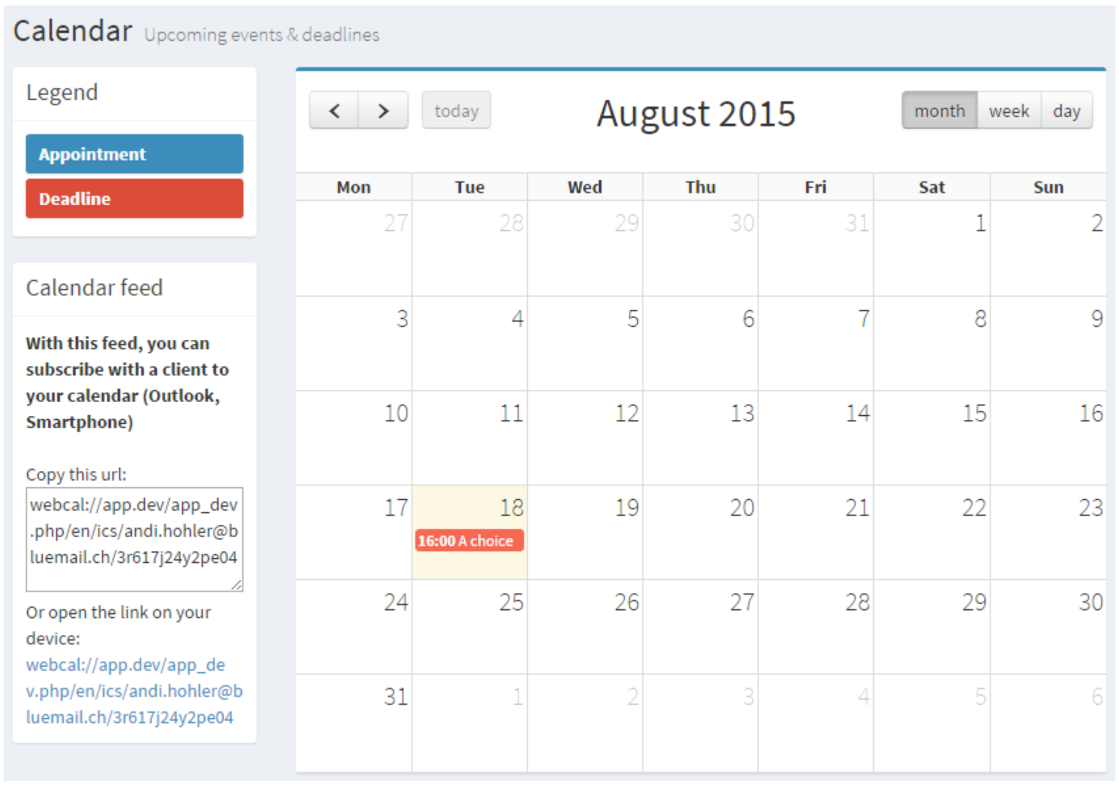
Generate a new one:

Generate new avatar

Change avatar

Figure 4.20: Edit profile 2

4.1.4.5 Calendar



With this feed, you can subscribe with a client to your calendar (Outlook, Smartphone)

Copy this url:

webcal://app.dev/app_dev.php/en/ics/andi.hohler@bluemail.ch/3r617j24y2pe04

Or open the link on your device:

webcal://app.dev/app_dev.php/en/ics/andi.hohler@bluemail.ch/3r617j24y2pe04

Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18 16:00 A choice	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Figure 4.21: Calendar

We implemented a simple calendar[17] that displays all poll deadlines and the fixed dates of schedule polls. The calendar supports several views (month, week, day).

A nice feature is the iCal calendar feed, implemented with the help of IcalBundle[18] for Symfony. The user can subscribe to it on his Phone or with his Mail program and the dates are automatically loaded from the website periodically (Period is depending on the used client and its settings). With this, a poll participant will never forget a deadline or appointment again.

This was not a risky feature, but also not that easy to implement because iCal feeds were new to us. We had some problems with getting a stream readable by all devices due to some semantic problems.

[17]`http://fullcalendar.io`
[18]`https://github.com/BorisMorel/IcalBundle` 21.07.2015

4.2 Architecture

We need a proper webserver environment with Apache[19] and MySQL[20] as a database management system. There is also access to a console needed to set up the Symfony environment. An installation guide for the webserver environment and Symfony can be found in the "Anleitung zu wissenschaftlichen Arbeiten" (Appendix A).

These decisions are based on the requirements listed in *Chapter 3* about the technologies. We stated that as programming language PHP should be used. So a logical conclusion is to use the standard webserver (Apache2) combined with the apache2-php5 module in order to get the best compatibility with web providers.

The justification of choosing Symfony as framework can be found in the *Appendix C: Technologies evaluation*. We compared it to another framework and made the decision based on our requirements.

Users interact with the front end by creating and using polls. The back end processes the user input, handles requests, manages the stored data and more. The back end relies on the MVC (Model View Controller) framework Symfony2[21] which provides several services like a User Management System, the Doctrine[22] ORM extension and the Twig[23] template engine. Our created templates are based on the Twig syntax and are parsed by the Twig template engine. It compiles the template to an HTML view with all included variables. The system is based on a MySQL database. The front end has dependencies on Twitter Bootstrap framework[24] and several other JavaScript and CSS extensions which provides multiple additional functionalities.

[19]`http://httpd.apache.org/`
[20]`https://www.mysql.com`
[21]`https://symfony.com/`
[22]`http://www.doctrine-project.org/`
[23]`http://twig.sensiolabs.org/`
[24]`http://getbootstrap.com/`

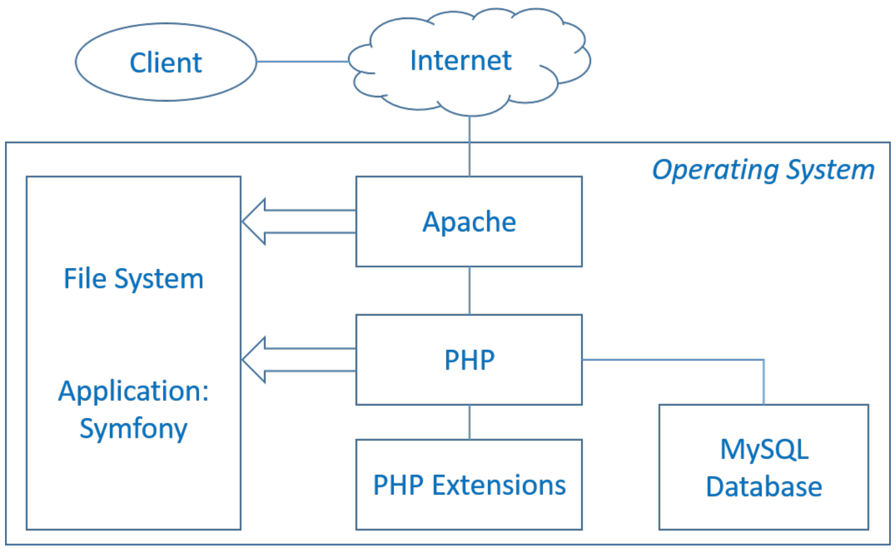


Figure 4.22: Webserver environment

4.2.1 Symfony

In this section we demonstrate how we use specific parts of Symfony in our application or how they work in order to better understand further explanations.

4.2.1.1 Request flow

Using the poll view as example, the Symfony request flow generally looks like this:

Request

Everything begins with a request. Symfony takes the request to determine if there is a matching route in the system defined. To get this working, we tell Symfony, which routes we want to match with specific controllers of the MVC structure and their functions. The route matching happens in the Kernel.

Kernel

The important things happen here. Symfony takes the URL and splits it up in pieces. The relevant pieces are tested against the defined routes. If there's a matching route, the Kernel will load the associated controller and call the right function.

[25] `http://code.tutsplus.com/tutorials/diving-into-symfony-2--net-32923` 18.08.2015

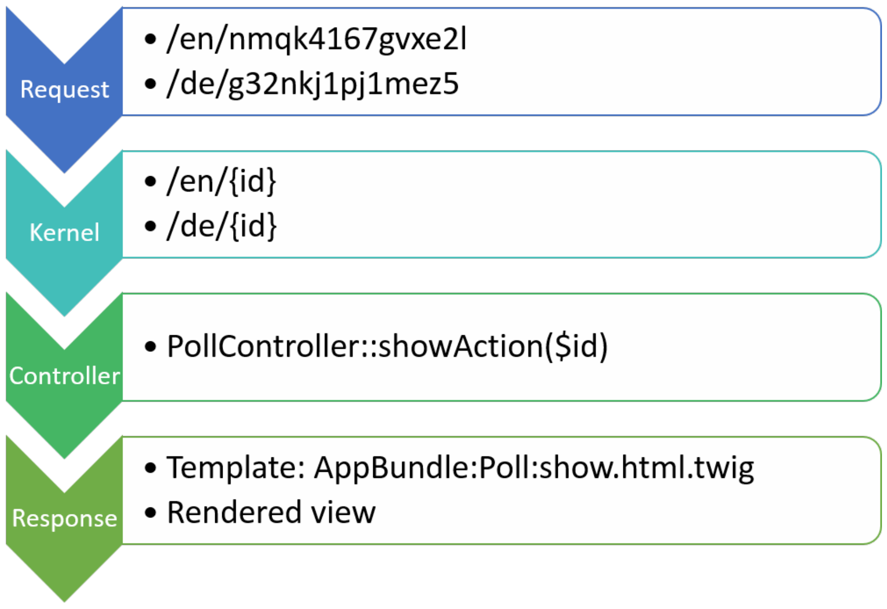


Figure 4.23: Symfony request flow[25]

Controller

The controller is loaded by the Kernel and a given action (defined as a function) is executed based on the route.

Response

A Symfony request must return a valid response object. This object can be put together in different ways, for example with or without headers. It can contain payload formatted as json, HTML or something else. If an action does not return a valid response, an exception will be thrown.

4.2.1.2 Basic file structure

```
app/
src/
vendor/
web/
```

The app directory

Here goes the project's global configuration. It's the home directory of the AppKernel.php which loads all relevant classes, third-party libraries and bundles into the framework. The folder also contains some utility classes.

The src directory
The “src” folder contains bundles with our code. A bundle is a logical group for Symfony. Bundles should be plug-and-play, so they can be used in every Symfony project without adapting the system to it. For example, a generic user system is outsourced in a bundle, so it does not depend on other components that don’t belong to the bundle.

There are many ready-to-use bundles for Symfony on the web. The website “KNP Bundles”[26] is a database containing many bundles and developers.

The vendor directory
In this directory all third-party libraries are stored. There are already many libraries, for instance Symfony, Doctrine (Object Relational Mapping), Assetic (for processing client side scripts), Swiftmailer (for sending Emails), phpUnit (for unit testing) and more. They get automatically updated by executing the composer update command.

The web directory
The web directory should be the root directory of the domain, because it’s the only folder of the project that is publicly accessible. In this directory we can find the basic controllers `app.php` (production mode) and `app_dev.php` (development mode). The app_dev.php disables caching and shows a developer bar with information about the current state of the Symfony application.
App file structure
The `AppBundle` is our bundle that contains the core of the website including several subfolders like Component, Controller, DataFixtures, and more. A short explanation:
Component: Additional classes used in the project
Constants: A static class with constants
Controller: All controllers that are used by the system.
DataFixtures: Contains initial entity objects that are loaded into the database by the command `php app/console doctrine:fixtures:load`
DependencyInjection: Modify Symfony settings at runtime.
Entity: Database entity classes
EventListener: Listener that are called on specific events.
Form: Form elements
Resources: This folder contains configuration files, Twig[27] templates, CSS & JavaScript files and the translation files.
Security: The object Voters are located here. They decide, whether a user is allowed to access/modify an entity or not.
Tests: phpUnit tests
Twig: Some Twig addons

[26] `http://www.knpbundles.com/`
[27] Explained in section 4.2 Architecture

ThemeBundle: Overrides some parts of the used theme AvanzuAdminTheme

```
app/
config/
\app\Resources
\src
\src\AppBundle
\src\AppBundle\Component
\src\AppBundle\Constants
\src\AppBundle\Controller
\src\AppBundle\DataFixtures
\src\AppBundle\DependencyInjection
\src\AppBundle\Entity
\src\AppBundle\EventListener
\src\AppBundle\Form
\src\AppBundle\Resources
\src\AppBundle\Security
\src\AppBundle\Tests
\src\AppBundle\Twig
\src\ThemeBundle
\src\ThemeBundle\Controller
\src\ThemeBundle\Event
\src\ThemeBundle\Model
\src\ThemeBundle\Resources
\vendor
\web
```

4.3 Technical implementation

4.3.1 Routing

We defined a controller for every sub-page (like Poll, Group, Rank, Calendar, Profile, etc.) where it handles page-specific actions like creating, updating, deleting and showing an entity. A route is a map from a URL path to such a controller. For example, we want to match any URL like /poll/2355urw68erafj or /poll/at6ke3r7j334cy and send it to a specific controller. The route can be defined with a simple annotation:

```php
// src/AppBundle/Controller/Poll/PollController.php
namespace AppBundle\Controller\Poll;
use AppBundle\Controller\MyController;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
/**
 * @Route("/poll")
*/
class PollController extends MyController {
  /**
   * @Route("/{id}", name="poll_show", requirements={
```

```
   *     "id": "[a-z0-9]{14}"
   * })
   * @Template("AppBundle:Poll:show.html.twig")
   */
  public function showAction(Request $request, $id) {
  // ...
  }
}
```

First, a route "/poll" for the whole class is defined. Alternatively this would be implemented in the `src/AppBundle/Resources/config/routing.yml` file. We define the route "poll_show" with a parameter "id" that should contain letters and numbers with a length of 14. The defined path acts like a wildcard "/poll/*" where the wildcard is given the name "id". The name "poll_show" must be unique and is the internal name. With it, we will be able to generate links to that specific route. There is also the possibility to define routes in external files in languages like YAML, PHP or XML. The goal of the Symfony routing system is to parse a given URL and determine which controller should be executed. The process is shown in the following image.

1. The request is handled by the Symfony front controller.
2. The Symfony Kernel asks the router to inspect the request.
3. The router matches the incoming URL to a specific route and returns information about the route, including the controller and method that should be executed.
4. The Symfony Kernel executes the method of the given controller, which ultimately returns a Response object.

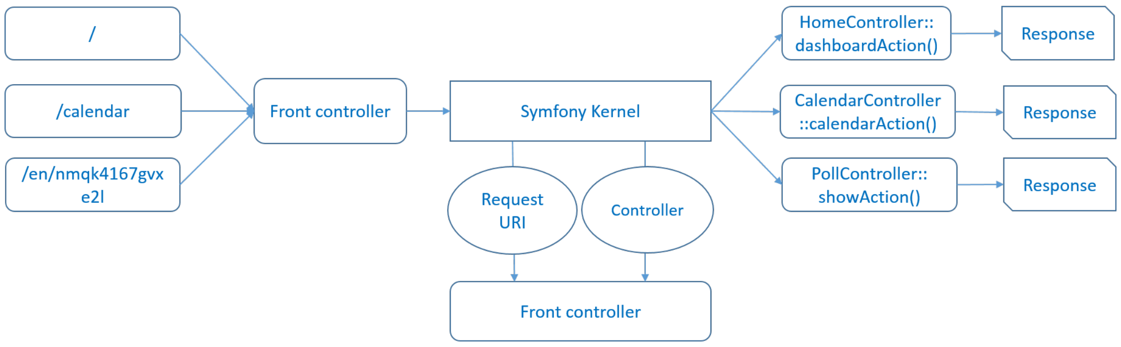


Figure 4.24: Symfony routing flow

Challenge

Besides that multiple languages were not a requirement, we thought that already implementing this functionality would support further extensions. The challenge here was to implement the language pattern to have first "/de/" or "/en/" in the URI[28], and then followed by the controller paths. It's not enough to just specify this parameter, because somewhere the language must be set in the system. We could solve this by specifying the "app" route parameter for our application bundle in the global routing file. The "root" definition handles requests without any parameter and redirects them to a controller that determines the language based on some browser parameter.

```
root:
    pattern: /
    defaults:
        _controller: AppBundle:Home/Default:redirectLanguage

app:
    resource: "@AppBundle/Resources/config/routing.yml"
    prefix:   /{_locale}
```

4.3.2 Security

Symfony provides a powerful security system that also contains an authentication extension. With this, we can easily set up a role-based login and a user management system.

Authentication

We can define different authentication providers and create firewalls for routes in the configuration file app/config/security.yml (global configuration over all bundles). The firewalls are the heart of our security configuration.
In a firewall, a request can be matched against routes, hosts, HTTP methods, pattern or other details of the request. The security is highly configurable and there are even more options we'll not use here. Our authentication part of the security configuration looks like this:

```
security:
    # the main part of the security, where you can set up firewalls
    # for specific sections of your app
    firewalls:
        # disables authentication for assets and the profiler
        dev:
            pattern:  ^/(_(profiler|wdt)|css|images|js)/
```

[28] https://en.wikipedia.org/wiki/Uniform_Resource_Identifier

```
            security: false
        #Default AppBundle
        main:
            pattern:    ^/([a-z]{2}/)?.*
            remember_me:
                key:       "%secret%"
                lifetime: 31536000 # 365 days in seconds
                path:      /
                domain:    ~ # Default: current domain from $_SERVER
            anonymous: ~
            http_basic: ~
            form_login:
                login_path: login
                check_path: login
                csrf_provider:  form.csrf_provider
                csrf_parameter: _csrf_token
                intention:                        authenticate
                username_parameter:               _username
                password_parameter:               _password
                success_handler: login_listener
            logout:
                csrf_parameter:       _csrf_token
                csrf_token_generator:  form.csrf_provider
                csrf_token_id:        logout
                target:               /
                path:                 logout
                success_handler: logout_listener
                invalidate_session: false
            switch_user: { role: ROLE_SUPER_ADMIN }
```

We defined that per default all sites are behind the "main" firewall. Also, the Remember-Me feature is enabled and login & logout target controllers are specified enabling us to show custom success/failure messages. The "switch_user" key allows us to impersonate other accounts, if the role "ROLE_SUPER_ADMIN" is assigned to our account. We'll have a further look into roles in the next section.

We define that the login controller is excluded from the firewall and the logout page requires a fully authenticated user:

```
security:
    # with these settings you can restrict or allow access for
   different parts
    # of your application based on roles, ip, host or methods
    access_control:
        #Exclude login form
        - { path: ^/auth, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        #Special rule for logout page: user must be logged in
        - { path: ^/auth/logout, role: !IS_AUTHENTICATED_ANONYMOUSLY
    && IS_AUTHENTICADED_FULLY }
```

The following shows the user provider properties. We define that the user password is encoded by the bcrypt hash function. It's based on the blowfish algorithm and a cost variable (standard value: 13) can be defined.

```
security:
    encoders:
        AppBundle\Entity\User:
            algorithm: bcrypt
            #cost: 13
    providers:
        users:
            entity: { class: AppBundle:User }
```

We had some issues with the authentication when it was not correctly authenticating the user in connection with the email address. We solved this problem by renaming the "email" field in the User entity to "username" and adding a custom implementation for the entity repository where we explicitly create a sql query statement to get the user entity.

Authorization

This is the part where we allow or deny access to specific roles and work with User objects. We configure the role hierarchy also in the security.yml file.

```
security:
   role_hierarchy:
        ROLE_MOD:          ROLE_USER
        ROLE_ADMIN:        [ROLE_USER, ROLE_MOD]
        ROLE_SUPER_ADMIN: [ROLE_USER, ROLE_MOD, ROLE_ADMIN,
    ROLE_ALLOWED_TO_SWITCH]
```

Used roles don't need to be defined anywhere, we can just start using them. There are already some default roles like "IS_AUTHENTICATED_ANONYMOUSLY" (user is not logged in), "IS_AUTHENTICATED_FULLY" (user is logged in) or "IS_AUTHENTICATED_REMEMBERED" (user was logged in automatically with the remember me cookie). The process of authorization has two different sides:

1. The user receives a set of roles when logging in.
2. Add code so that a resource requires a specific attribute (role) in order to be accessed.
    - Use access_control in security.yml, which allows URL patterns to be protected (e.g. /private/*). This is easy, but less flexible.

- In the code with the security.authorization_checker service.
- As annotation for controller functions.

For controllers, we use annotations provided by the SensioFrameWorkExtraBundle:

```
// ...
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;

/**
 * @Security("has_role('ROLE_ADMIN')")
 */
public function helloAction($name)
{
    // ...
}
```

In Twig templates, the access control works like this:

```
{% if is_granted('ROLE_ADMIN') %}
    <a href="...">Delete</a>
{% endif %}
```

4.3.3 User permissions on objects

How can we handle permissions for object manipulation in a simple way? We could easily check the permission like this:

```
if($object->getOwner()->isEqualTo($this->getUser()) {
...
}
```

But that's not how Symfony wants you to do it, so we could check permission to access data by using the ACL (access control list) module. Since this is a bit overwhelming for our application we instead use a much easier solution which is to work with custom voters for all necessary entities (Group, Poll, Comment, PollEntry, Notification). Voters are like simple conditional statements. A voter decides, if the current user should have access to a specific resource. All existing voters in Symfony are called everytime we use the isGranted($attribute, $object, $user = null) function. In a voter, we define attributes and the supported object class (mostly an entity). Here is a simple example of the CommentVoter implementation we use for comments in polls:

```
class CommentVoter implements VoterInterface {
  // define attributes
  const VIEW = 'view';
  const EDIT = 'edit';
  const DELETE = 'delete';
```

```
public function supportsAttribute($attribute) {
  return in_array($attribute, array(
    self::VIEW, self::EDIT, self::DELETE
  ));
}

public function supportsClass($class) {
  $supportedClass = 'AppBundle\Entity\Comment';
  return $supportedClass === $class || is_subclass_of($class,
 $supportedClass);
}

public function vote(TokenInterface $token, $comment, array
 $attributes) {
  // check if class of this object is supported by this voter
  if (!$this->supportsClass(get_class($comment))) {
    return VoterInterface::ACCESS_ABSTAIN;
  }

  // set the attribute to check against
  $attribute = $attributes[0];
  // check if the given attribute is covered by this voter
  if (!$this->supportsAttribute($attribute)) {
    return VoterInterface::ACCESS_ABSTAIN;
  }
  // get current logged in user
  $user = $token->getUser();
  // make sure there is a user object (that the user is logged in)
  if (!$user instanceof UserInterface) {
    return VoterInterface::ACCESS_DENIED;
  }

  switch($attribute) {
    case self::VIEW:
              return VoterInterface::ACCESS_GRANTED;
      break;
    case self::EDIT:
              if ($user->isEqualTo($comment->getUser())) {
                  return VoterInterface::ACCESS_GRANTED;
              }
      break;
    case self::DELETE:
      if($user->isEqualTo($comment->getUser())) {
        return VoterInterface::ACCESS_GRANTED;
      }
      break;
  }

  return VoterInterface::ACCESS_DENIED;
  }
}
```

At first, Symfony didn't recognize the voter. Then we discovered to get the voter working, Symfony must inject it into the security layer. This means we have to declare it as a service and tag it with security.voter.

```
# app/config/services.yml
services:
    security.access.comment_voter:
        class:      AppBundle\Security\Authorization\Voter\
    CommentVoter
        public:     false
        tags:
            - { name: security.voter }
```

4.3.4 Translation

The whole application is multi-lingual and translated to English and German for consistency reasons. This is done by defining translations of strings in external YAML files. We can work with translation groups to keep the translation folder structured. To translate a string in a template, we use the following code fragment:

```
{{ 'Your name is: %name%'|trans({"%name%": user.name}, 'translation-
    group') }}
```

There is also a fallback mode available when no translation exists. Either it switches to the specified fallback language or just displays the string we try to translate. It's also possible to use a translation schema like ”Account.Overview.welcome”. The yaml definition in the translation file would then look like this:

```
Account:
   Overview:
      welcome: Your name is: %name%
```

4.3.5 Database

The Symfony framework doesn't integrate any ORM (Object-relational mapping) by default, but the Symfony Standard Edition comes bundled with Doctrine. Doctrine aims to give powerful tools to make the implementation of databases easy. It lets us map objects to a relational database such as MySQL, PostgreSQL or Microsoft SQL. It's also possible to use MongoDB with Doctrine.

Doctrine allows us to not just fetch rows of a column-based table into an array, but to persist entire objects to a database and fetch entire objects out of the database. This works by mapping a PHP class to a database table and the properties of that PHP class to columns on the table.

Since Doctrine dictates based on the entity definitions how the database is built up, we can't really modify that structure.

Configuration

We configure the database connection parameters in the app/config/parameters.yml file.

```
# app/config/parameters.yml
parameters:
    database_driver:    pdo_mysql
    database_host:      localhost
    database_name:      project
    database_user:      root
    database_password:  password
```

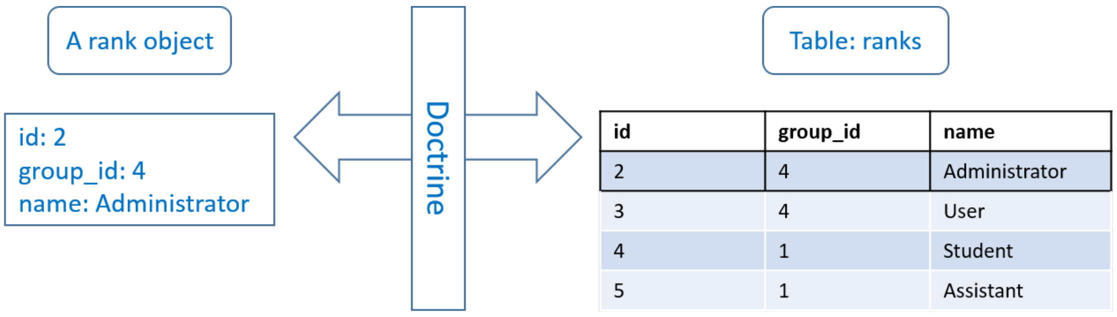
Entity class

Now we can create an entity class that holds data and can be persisted. Just add class variables as ususal. To add mapping information for Doctrine, we use the @ORM-Annotations to add metadata.

```
// src/AppBundle/Entity/Rank.php
namespace AppBundle\Entity;
/**
 * @ORM\Table(name="ranks")
 * @ORM\Entity()
*/
class Rank {
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;
    /**
```

```
      * @ORM\ManyToOne(targetEntity="Group", inversedBy="ranks")
      * @ORM\JoinColumn(name="group_id", referencedColumnName="id")
      */
     protected $group;
     /**
      * @ORM\Column(type="string")
      * @Assert\Type(type="string")
      */
     protected $name;
}
```

With the mapping metadata, Doctrine knows how to handle the whole thing:



id	group_id	name
2	4	Administrator
3	4	User
4	1	Student
5	1	Assistant

Figure 4.25: Rank entity example

To generate all necessary setters and getters, Doctrine provides a nice command:

```
php app/console doctrine:generate:entities AppBundle/Entity/Rank
```

And with the following command it updates the database schema:

```
php app/console doctrine:schema:update --force
```

The library automatically generates tables based on the entity objects. And actually, this command is very powerful. It compares what the database should look like (based on the mapping information of the entities) with how it actually looks, and then generates the SQL statements needed to update the database to the wanted state. In other words, if we add a new property with mapping metadata to our Rank class and run this task again, it will generate the “alter table” statement needed to add that new column to the existing table.

An other way to take advantage of this functionality is via migrations, which allow us to generate these SQL statements and store them in migration classes that can be run systematically on the production server in order to track and migrate the database schema safely and reliably.

User account

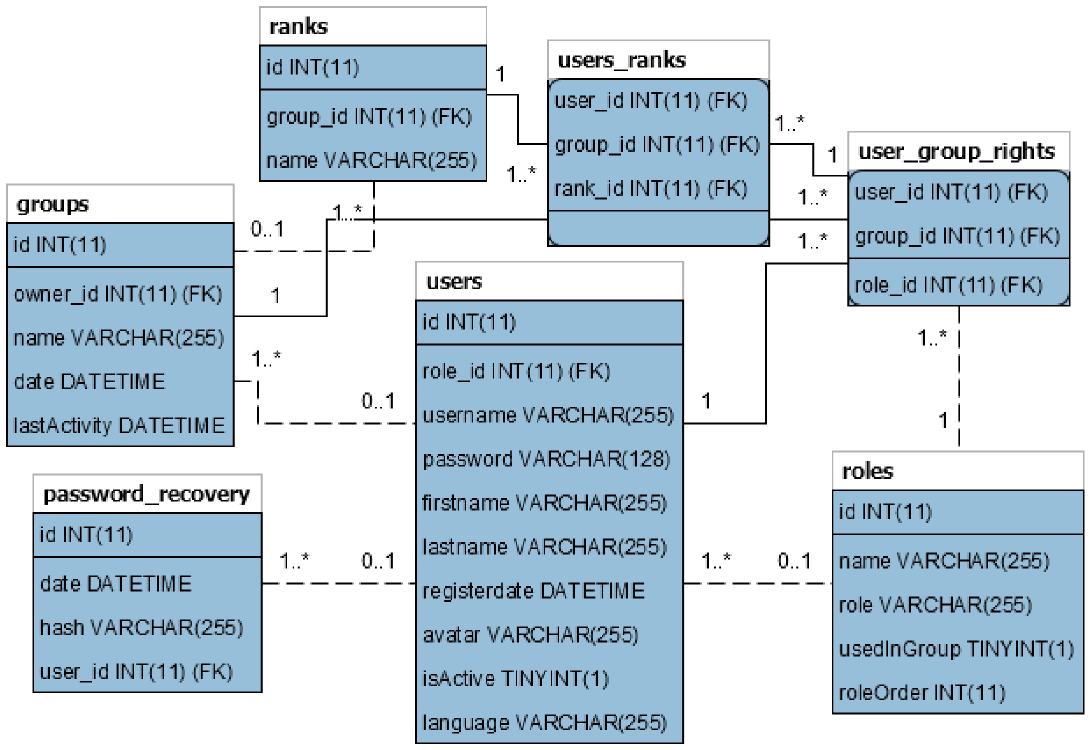


Figure 4.26: User account tables

The users table in the middle represents the core table of the user system. It holds all information directly related to users such as username (email address), role, name and password. It's related to groups over a special join table that also contains the specific role of a user in the connected group. We have created 4 roles:

Name	Role	Description
User	ROLE_USER	Default role after registration and in groups
Moderator	ROLE_MOD	Moderator in a group
Group-Administrator	ROLE_ADMIN	Group owner
Administrator	ROLE_SUPER_ADMIN	System administrator, can impersonate other users

The “usedInGroup” flag tells the system if a role is also used in groups.

Poll

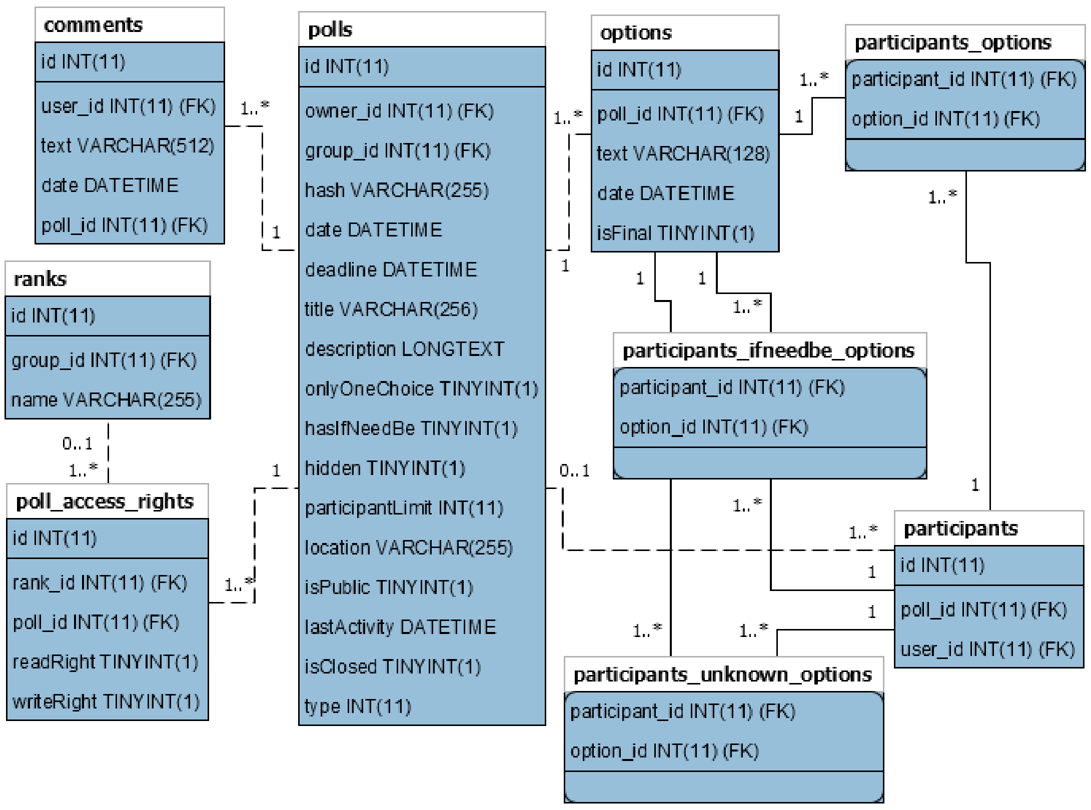


Figure 4.27: Poll tables

This constellation is a bit complicated. The poll table holds the poll configuration like title, description, participant limit, etc. It has relations to poll options, comments, access rights and participants. The options themselves are connected to participants over the participants_options join table. ifneedbe options are also stored and obviously depend on options and participants. If new options are added to a poll, it creates unknown options for every participant. So the user sees if a poll is changed and if a participant already updated his entry.

The access rights are linked to the group ranks. This way we will be able to restrict access to polls inside groups.

Notification & Log

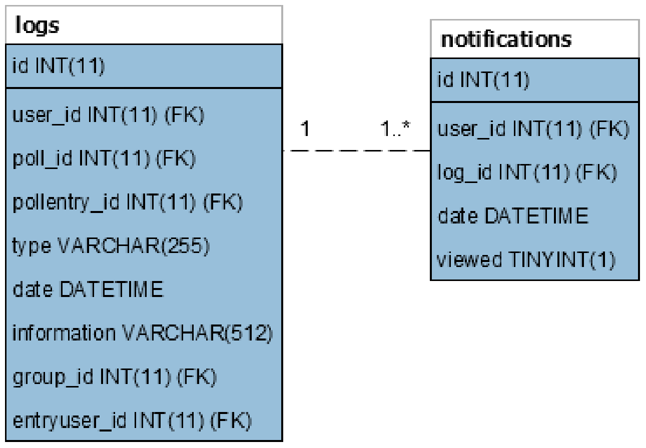


Figure 4.28: Notification & Log tables

We create log entries for several actions to make notifications easier. We define several types of logs for groups and polls:

Group types		**Poll types**	
Variable	Type	Variable	Type
CREATE	group_create	CREATE	poll_create
EDIT	group_edit	INSERT	poll_insert
DELETE	group_delete	INSERT	poll_insert
LEAVE	group_leave	EDIT	poll_edit
CHANGE_NAME	group_change_name	REOPEN	poll_reopen
CHANGE_OWNER	group_change_owner	ENTRY_EDIT	poll_entry_edit
MEMBER_IMPORT	group_member_import	CLOSE	poll_close
MEMBER_ADD	group_member_add	FINAL_OPTIONS	poll_final_options
		COMMENT_ADD	poll_add_comment
		COMMENT_DELETE	poll_delete_comment

5 The Validation

It's challenging to validate that this software project solves the initially discussed problem. In the first part, we do some usability tests to improve the user experience and find bugs and flaws. In the second part, a Qualitative Case Study is carried out.

5.1 Usability tests

We made a few usability tests during the development period with different study participants and a light outline. Some are Computer Science students, some are non-informaticians. It's interesting how the different people are aware of details.
We defined some use cases which should cover most of the application.

5.1.1 First test

The first usability test was done in May 2015 with a Computer Science (CS) student and a pharmacist (PH).
1. Sign up
CS and *PH* suggested that the sign up button should be more exposed. *CS* gave some general design advices. All suggestions were implemented later.
2. Group creation
Both stated that the link to create a new group is hard to find. It's hidden in the navigation under Groups. *CS* criticized the wizard; the layout is not optimal and is in general poorly designed. We got the message and implemented a much better wizard with a nicer design

and no page-reloading after every step.
CS commented that in the group list, the group id should be removed. Also, the group name should be directly linked to its group dashboard. Finally, he suggested that if there is no group, the information message should not be red due the fact that this color usually indicates an error and not an information. The suggestions were adopted and the information message color was changed later to blue.
3. Create a poll in the newly created group
CS again noticed some details in the design and made a recommendation which we followed.
PH didn't find a button to create polls in the group list. This was fixed later. *PH* missed some information in the wizard about the poll settings. In this phase of the project it was simply not yet added.
4. Make a poll entry
CS suggested to change the color of informational text from red to another (blue), which was done later. *CS* thought that it would be nice if his own poll entry is highlighted to better see it. This was also implemented.
5. Edit own poll entry
PH found a bug: if he double-clicks on the edit button, the edit form shows up twice. This was fixed shortly after it was discovered.
6. Close poll No problems occurred.
7. Reopen poll No problems occurred.
8. Edit poll & set new deadline
PH found two bugs. Even if in the creation process a deadline is specified, in the edit form the deadline button is not checked and the calendar is not shown.
The second bug was that existing ifneedbe options were not deleted, if the ifneedbe setting was disabled later. Both bugs could be fixed.
PH found it not very practical that the tabbing order of the Option fields were like this: Input field - Delete button - Next input field. It should just jump from one input to the next input field. We thought this was a good improvement for the usability.

9. Add a member to the group
PH stated that the Save & Back buttons are in a wrongly appearing order. *PH* was right and we switched them.
CS would liked to have a settings button in the group dashboard to access important group settings. *CS* mentioned some minor design aspects we then improved.

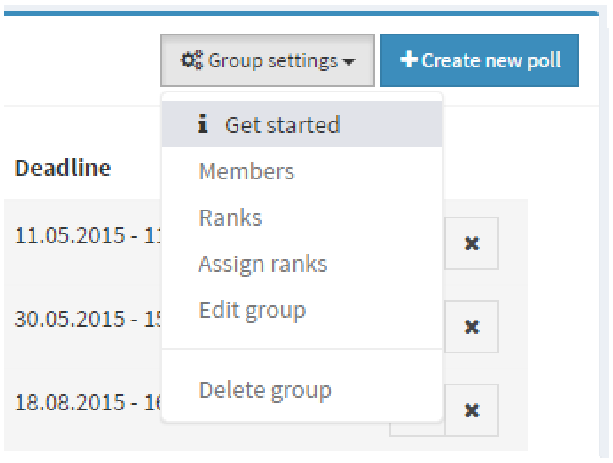


Figure 5.1: Settings button in group dashboard

10. Add ranks
PH suggested a better order of the buttons on the form bottom.
11. Assign ranks
CS found it confusing because the member management layout looks the same, and he didn't see at first the difference between ranks and roles.

12. Edit profile
CS thought that we should rename "deactivate account" into "delete account". We technically can't completely delete an account because otherwise there would be problems all over the application.
We solved it with displaying a deleted account information on login attempts with deactivated accounts.
PH suggested that the Profile button should be more highlighted. It was justified and we adapted it (Figure 5.2).

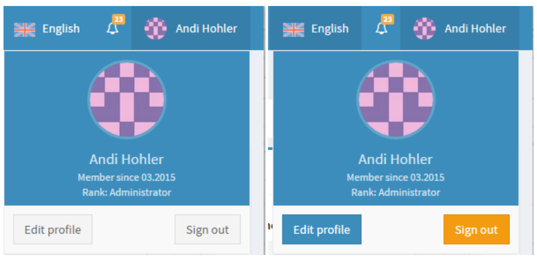


Figure 5.2: Before and after: Edit profile & Logout

13. Edit poll entries of other users
PH saw that the last activity of the poll didn't get updated after editing an antry. This was fixed.
General
CS mentioned some more things:

- The "Friends" button in the user dropdown had no functionality
- Some spelling failures
- Hide whole tables if there is no content
- Remove red fonts where it's just an information and not an error
- Overloaded tables in dashboards
- Missing legend for tables
- Add possibility to change the group owner
- some more design aspects

We tried to improve the mentioned points as much as possible.

5.1.2 Second test

This second test was done mid May and June 2015 with another Computer Science student (CS) and a bank employee (BE).

1. Sign up

CS thought we should implement third party authentication from Facebook and/or Google. It was not a requirement but would be a good feature for future improvements. He mentioned that Internet Explorer 9 doesn't support the "placeholder" attribute of form elements. We thought that most of today's internet users don't use Internet Explorer 9 anymore. We just focused on current versions of Mozilla Firefox[29], Google Chrome[30] and Microsoft Internet Explorer[31]. Only if CS is using the sign up form the Google Captcha reCAPTCHA service thinks he's a bot and displays the most annoying captcha. He was not amused.

2. Create a group

BE noticed that a group name must be unique. CS wanted the blue success message to be green.

3. Add members to the newly created group

On advice of *CS* we added more information to the success page.

4. Add ranks to the group

CS suggested to add some empty fields if there are no ranks specified yet. He also thought we should change the "Remove" button to a "X" button because it's more intentional. He also found a bug: In the remove dialog was just a placeholder for the rank name instead of the real name. We fixed and improved all mentioned things.

5. Assign ranks to members

No problems occurred

6. Create a text poll in the group

CS suggested to change some texts and add better description to the poll settings. Also he wanted to have tooltips for icons in the poll view. He found a bug that the Ranks tab in the poll wizard was not displayed. Sometimes if the internet connection is bad, and the site (only the poll wizard) rests white for a short moment. This is because the Google Places API is loaded before the context. We fixed the problems and added the features.

7. Make a poll entry

CS also noticed that the order of the "Save" & "No option fits" buttons is not intentional. We changed it.

8. Edit own poll entry

No problems occurred.

9. Edit the poll

SC suggested to again change the "Remove" to a "X" button in the Option wizard step.

[29]`https://www.mozilla.org/en-US/firefox/new/`

[30]`https://www.google.com/chrome/`

[31]`http://windows.microsoft.com/en-US/internet-explorer/download-ie`

He noticed that if the “only one option per user” setting is enabled, entries with more than one positive option rest the same. We thought that this should not change. *BE* found a bug: a JavaScript variable was not defined.

10. Fix final options

CS mentioned that he should be able to reopen the poll when it already has fixed options without setting a new deadline or if the deadline is still in the future. He suggested that the form to choose final options should not disappear if the deadline is reached. Also he liked to have the final options highlighted in the view.
We followed his suggestions and implemented all of them.

11. Create schedule poll

CS stated that the description where he could choose between text and time poll is not sufficient. He noticed that the calendar had a wrong localization setting. The participant found a bug: If the group was changed multiple times in the poll wizard, the available ranks got appended to the content instead of changed.
All this was fixed.

12. Make a poll entry

No problems occurred.

13. Edit own poll entry

No problems occurred.

14. Edit the schedule poll

No problems occurred.

15. Fix final dates in poll

No problems occurred.

16. Calendar

The calendar had a wrong localization setting. Entries are shown multiple times in day and week view. We found out that this is a bug in the JavaScript plugin the theme uses, and we can’t fix it by ourselves the time this test was accomplished. Also, other people’s events were shown. This was a bug and was fixed soon.

17. Edit profile

No problems occurred.

General

BE thinks it’s good if he can reactivate his deleted account over Email. Also there was a bug in the GUI where buttons in the dropdown navigation were transparent instead of blue.

5.2 Qualitative Case Study

The User Story

The user wants to organize a chess team event at the University of Bern with a maximum of 20 participants. It must have several date possibilities. Also on every date, there should be two time options: in the morning and afternoon. All chess team members (Juniors, Seniors) are invited. The poll must be closed one week before the first date. The user wants that people can see all entries. He can only use our application to complete this task.

The Study Participant

Linus Schwab, a Computer Science student, agreed to take part in this test.

The Registration

First, Linus tried to bypass our application and started to navigate to doodle.com for completing the task. We got the joke and reminded him of the task conditions.
He had no problems to sign up. He inserted his first and last name, added the email and used a secure long password.

The Group

Linus created a chess group and noticed that he has to add the necessary ranks Juniors and Seniors to properly organize the group. He didn't see the save button at the bottom, so he expected that the form for creating ranks would auto-save.
Then he added some members and assigned them to different ranks (Juniors and Seniors).

The Poll

He had no problems finding a way to create a new poll. In the second wizard step, he did not realize at first that he can copy the first date row to all other rows. But then he noticed that it is a very useful function. Linus correctly remembered that the deadline is exactly one week before the first date, and chose it accordingly.
The user thinks that in the fourth wizard step, it should be more descriptive and the permissions should not be named "Read" and "Write", but "Can participate in the poll" and "Can see the poll".
Now the poll was created successfully and invited people will participate in the poll.
Linus also participated and chose his favourite options. There didn't occur any problems at all.

The End

Now that the deadline is reached, the poll is closed automatically. He saw that he can fix the final poll options, was pleased of this feature and used it. The test participant used the most popular option as decision help and fixed the poll with this date.

Conclusion

Our application convinced Linus and he would use it as soon as it's released for the public. He made several suggestions to make the tool even better. We considered his thoughts and improved the application.

6 Future Work

We all know a software project is never finished and can be extended and improved infinitely. So here are some possible extensions discussed which we could implement after the thesis.

Third party authentication

To make the authentication easier and eliminate the need for the user to utilize another website account, we could implement third party authentication (with an OAuth client) using Google, Facebook or Twitter. This could be implemented with the HWIOAuth-Bundle[32] for Symfony which supports over 40 different providers.

Mobile Apps

We built an easy to use mobile-first web application. Now what would be perfect? Correspondent apps for mobile phones. We could easily wrap the website into an app and distribute it with help of Google's[33] and Apple's[34] app stores.

Each app store has its own rules. For example, if an app for iOS just wraps a website and the whole functionality could also be provided only through a mobile browser (e.g.

[32]`https://github.com/hwi/HWIOAuthBundle`
[33]`https://play.google.com`
[34]`http://www.apple.com/chde/itunes/charts/free-apps/`

Safari), it gets declined.

An additional nice feature are push notifications so the user won't miss any deadlines or new polls.

We decided to exclude these features (notifications, mobile app) despite listing them in the Software Requirements Specification.

Social platform

To keep up with the trend to make everything social we could implement a facebook-like wall in groups. There could show up group notifications so every user can see it (i.e. somebody has participated in a poll) and members can post things.

General

Furthermore, some customized views (i.e. poll view) for the mobile access could be developed to achieve a better user experience on small displays.

7 Conclusion

In this chapter we look back at the entire project and think about what we have achieved and what we could have done better.

The project

In the beginning, we put up many requirements. We learned during the project that there is a limit of things we will be able to deliver at the end of the project. This resulted in that we at last couldn't handle all wanted requirements. But that's not a defeat since we discussed some good ideas in *Future Work* we are going to work on after the thesis.

This was a relatively big project for us to master. It was very interesting and challenging to work with new technologies and to find a way to solve emerging problems. Sometimes we had to take a step back and begin with a new approach if a problem could not be solved the way we tried.

Result

We created a working nice looking application that can be used productively by everyone. We could fulfil all important requirements and have built a software to start polls in groups that was the intention of this thesis.

We implemented a proper authentication and authorization system. We tried to stick to the simplicity of Doodle when it came to the design of the poll system that includes displaying, creating and editing them.

Since the group system is part of the thesis' main goal, we are happy it's working as intended and supporting all required functions.

Personally, we would liked to have implemented the mobile applications too with a smooth notification system. Unfortunately, we had to put back these requirements in favour of finishing in time.

About used technologies

We learnt in a Computer Science Master course about Ruby on Rails[35] and how to build a web application with it. In the end, we would have seriously considered using a Ruby framework instead of a PHP framework. It's more lightweight and provides with gems (add-ons) an easy way to extend an application.

Also for future projects we would consider using a lightweight JavaScript front end and develop the back end as a restful API service to better split front and back end into independent applications.

Deployment

Most of our users tend to stick to familiar things like in this case Doodle and its mobile application. We decided to first finish the development of the mobile applications to successfully deploy the system for the sports club later without any migration flaws. Some key users that will mainly be responsible for polls are already testing the system in a closed environment. We already got several feedback from club members and other users that got their hands on the system.

It's planned to really finish this last part in the first quarter of 2016. This would just go with the ordinary General Meeting where we can officially announce the deployment.

[35]`http://rubyonrails.org/`

Appendices

Appendix A: Anleitung zu wissenschaftlichen Arbeiten

Contents

1 Getting Started with Symfony

In this tutorial we are going to show how to build a simple blog-like Symfony[1] application where we can create, show, edit and delete articles.

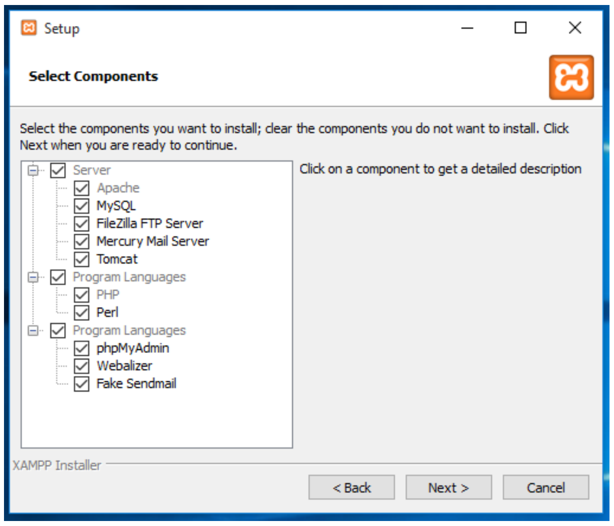
[1]`https://symfony.com/`

2
Setup & Configuration

We are going to use **Windows 10** for this tutorial. So all commands only apply on Windows. There is a different way if we would use Linux or Mac OS X.

2.1 Webserver

In order to use the Symfony installer, we have to install PHP[2]. There are two approaches. First is to install only PHP and use the new built-in HTTP-Server[3] of PHP 5.4. But we will also use XAMPP[4] because it provides a complete Webserver environment with Apache, PHP and MySQL[5].
We install XAMPP with all features into the directory C:\xampp.



Setup

Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer
  - ☑ Fake Sendmail

Click on a component to get a detailed description

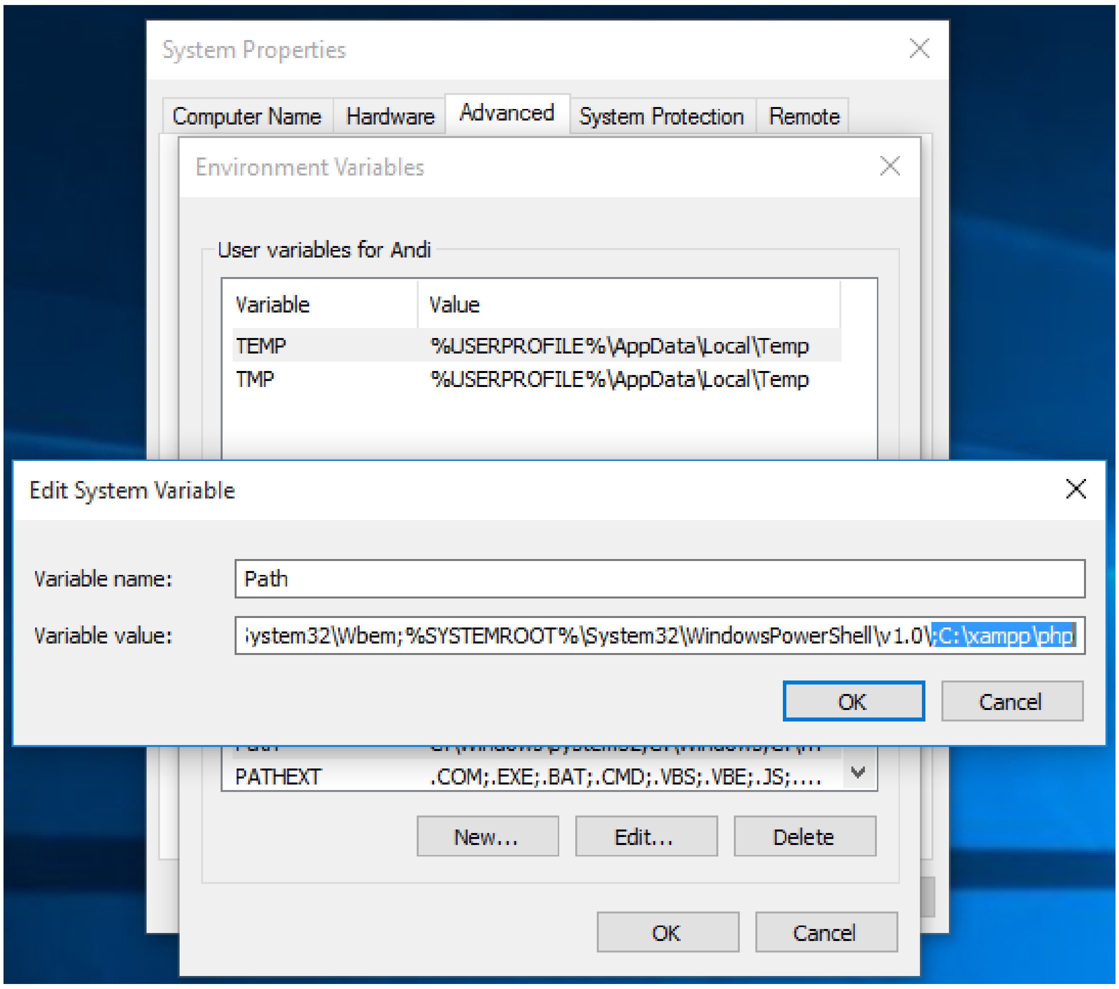
XAMPP Installer

Figure 2.1: XAMPP Setup

[2]http://php.net/
[3]http://symfony.com/doc/current/cookbook/web_server/built_in.html 14.09.2015
[4]https://www.apachefriends.org/index.html
[5]https://www.mysql.com/

Now we have to add PHP to the environment variables of the operating system in order to just use `php` as a command:

1. Use Windows+R to open "Run"
2. Type in SystemPropertiesAdvanced.exe and run
3. Click on "Environment Variables..."
4. Edit the variable "Path" in the system variables
5. Add `;C:\xampp\php` at the end and save



Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....

Variable name: Path

Variable value: ;ystem32\Wbem;%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\;C:\xampp\php

Figure 2.2: Environment Variables

So after installing the setup from XAMPP we go on with the Symfony setup.

2.2 Symfony

We open a windows console in the directory `C:\xampp\htdocs` and execute the following commands:

```
php -r "readfile('http://symfony.com/installer');" > symfony
php symfony new SymfonyTutorial
```

SymfonyTutorial is the project name you can choose by yourself. It can't contain white spaces. We could add a Symfony version as optional second parameter to the new command, if we want to explicitly specify the version. After the second command has finished executing, we see the following message:

```
Preparing project...

 OK  Symfony 2.7.3 was successfully installed. Now you can:

    * Change your current directory to C:\xampp\htdocs\
   SymfonyTutorial

    * Configure your application in app/config/parameters.yml file.

    * Run your application:
        1. Execute the php app/console server:run command.
        2. Browse to the http://localhost:8000 URL.

    * Read the documentation at http://symfony.com/doc
```

Now we change the directory to the newly created project.

```
cd SymfonyTutorial
```

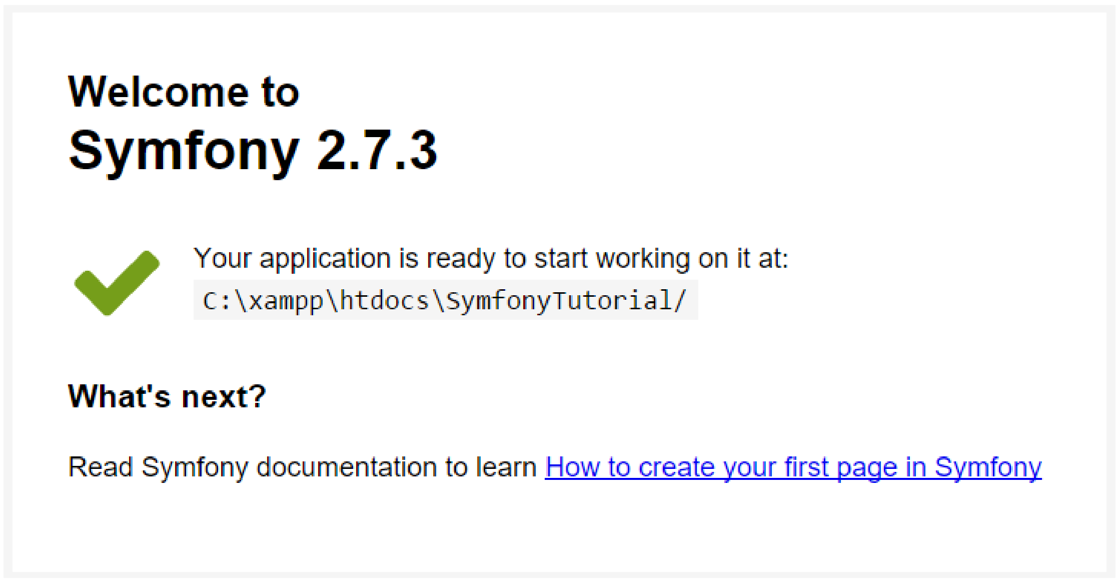
Brief description of the folder structure:

File/Folder	**Purpose**
app/	Global project configuration, log and cache files
src/	Bundles
src/AppBundle	Contains the controllers, models, views. Basically our application. We focus on this folder for the rest of this guide.
vendor/	Contains third-party libraries
web/	The only folder that should be accessible by public. Contains static files and compiled assets.
web/app.php	The file to access Symfony app in production mode
web/app_dev.php	This file is used in development mode and shows a debug toolbar
composer.json	This file allows you to specify what dependencies are needed for your application. This file is used by composer to load the libraries.

We start the built in webserver:

```
php app/console server:run
```

If we open thee URL `http://localhost:8000`, the following is displayed:



Welcome to
Symfony 2.7.3

Your application is ready to start working on it at:
`C:\xampp\htdocs\SymfonyTutorial/`

What's next?

Read Symfony documentation to learn How to create your first page in Symfony

Figure 2.3: Symfony start screen

We could also just fire up our XAMPP Apache server and open `http://localhost/SymfonyTutorial/web/app_dev.php`.

3 Getting Started

3.1 Say Hello

The Symfony setup created a default controller "src/AppBundle/Controller/DefaultController.php". But we won't use it. To get Symfony say Hello, we need to create at minimum a controller with a method and a view.
A controller's purpose is to receive specific requests for the application and collect information for the view, which purpose is to display this information in a human readable format. An important distinction to make is that it is the controller, not the view, where information is collected.
To create a new controller, we need to run the generator command[6] and tell it we want a controller called "Welcome" in the AppBundle bundle:

```
php app/console generate:controller --no-interaction --controller=
    AppBundle:Welcome
```

The result confirmed the controller creation:

```
Controller generation

Generating the bundle code: OK

You can now start using the generated code
```

[6] http://symfony.com/doc/current/bundles/SensioGeneratorBundle/commands/generate_controller.html 14.09.2015

It generated an extendable controller skeleton:

```
<?php

namespace AppBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;

class WelcomeController extends Controller
{
}
```

We have to manually create the method and its view. For the view, we create the folder src/AppBundle/Resources/views/Welcome/. In this folder, the following file must be created: index.html.twig

```
<h1>Welcome, Symfony</h1>
```

Now add the associated method to the Welcome controller. The route and template are defined with annotations. The index method should be called if we open “/” and it’s name in the router is “hello”. The template is located in “Welcome” in the AppBundle’s views folder and is named “index.html.twig”.

```
/**
 * @Route("/", name="hello");
 * @Template("AppBundle:Welcome:index.html.twig")
*/
public function index() {
  return array();
}
```

To disable the old route directing to the welcome page of Symfony, we just delete the DefaultController.php in our bundle.

4 Implementation

4.1 Create a controller with CRUD options

We want to have create, read, update and delete methods for our blog articles. Sized down to an acronym, this is called "CRUD" operations.
First, we have to generate[7] the entity file, on which the CRUD generator is based. The entity file is the representative class of the Article object. It's used by Doctrine[8] (ORM library in Symfony) to manage and persist data.

```
php app/console generate:doctrine:entity --entity=AppBundle:Article --
    fields="title:string(255) text:text" --no-interaction
```

The output will be something like this:

```
Entity generation

Generating the entity code: OK

You can now start using the generated code
```

This command creates the file `src/AppBundle/Entity/Article.php` which contains the title and text variables, and all necessary getters and setters. Also it generated the necessary annotations for Doctrine:

[7] http://symfony.com/doc/current/bundles/SensioGeneratorBundle/commands/generate_doctrine_entity.html 17.09.2015
[8] http://www.doctrine-project.org/

```
/**
 * Article
 *
 * @ORM\Table() //this class needs a table. With no parameter, it's
   named "articles"
 * @ORM\Entity  //this class is an entity
 */
class Article
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer") //table field
   definition: integer
     * @ORM\Id                    //auto_increment
     * @ORM\GeneratedValue(strategy="AUTO")   //type of
   auto_increment
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="title", type="string", length=255) //table
   field def.: string with length 255
     */
    private $title;

    [...]
}
```

Symfony provides one simple way[9] to create a controller that supports CRUD actions based on our new Article entity:

```
php app/console generate:doctrine:crud --entity=AppBundle:Article --
   with-write --no-interaction
```

This outputs

```
CRUD generation

Generating the CRUD code: OK
Generating the Form code: OK

You can now start using the generated code
```

[9]http://symfony.com/doc/current/bundles/SensioGeneratorBundle/commands/generate_doctrine_crud.html 17.09.2015

This command generates a bunch of methods in a new ArticleController.php file and also added its template files.

Methods/Actions

- indexAction - loads all articles and displays it
- createAction - creates a form to add new articles and handles the POST-Request
- newAction - displays a form to create a new Article entity
- showAction - finds and display an Article entity
- editAction - displays a form to edit an existing Article entity
- updateAction - edits an existing Article entity
- deleteAction - deletes an Article entity
- createCreateForm, createEditForm, createDeleteForm - creates the corresponding form

Templates

- index.html.twig - Table of all Article entities
- new.html.twig - Shows the create form
- edit.html.twig - Shows the edit form
- show.html.twig - View of a single Article entity

If we take a look on the method annotations, not every @Template annotation has an argument. This is because Symfony will take the method name {method}Action and looks up a template called {method}.html.twig in the controller-named folder in Resources/views/Article.

4.1.1 Checking router configuration

We can now check all routes in the application with a command and see that all necessary CRUD actions are correctly routed.

```
$ php app/console debug:router
[router] Current routes
Name                         Method Scheme Host Path
[...] (debug routes hidden)
article                      GET    ANY    ANY  /article/
article_create               POST   ANY    ANY  /article/
article_new                  GET    ANY    ANY  /article/new
article_show                 GET    ANY    ANY  /article/{id}
article_edit                 GET    ANY    ANY  /article/{id}/edit
article_update               PUT    ANY    ANY  /article/{id}
article_delete               DELETE ANY    ANY  /article/{id}
hello                        ANY    ANY    ANY  /
```

4.2 Database connection

Now we want to configure a database for our small application. We have two options: use the built-in MySQL from XAMPP or SQLite3[10]. SQLite3 just creates a database file, where MySQL is a whole DBMS[11] (Database Management System).
Both ways are described in the following two sub sections.

4.2.1 SQLite

We open the app/config/parameters.yml.dist file, add the database_driver parameter and set it to pdo_mysql (this is just the template we'll overwrite in the parameters.yml). Also it is needed to uncomment the database_path line:

```
database_driver: pdo_mysql                #add this line
database_path: "%kernel.root_dir%/data.db3"     #uncomment this line
```

"database_path" is the path to the SQLite database file. "kernel.root_dir" is the absolute path to the Symfony app directory.
Now we change the app/config/parameters.yml like this:

```
parameters:
    database_driver: pdo_sqlite
    database_host:
    database_port:
    database_name: %kernel.root_dir%/data.db3
```

[10] https://www.sqlite.org/
[11] https://en.wikipedia.org/wiki/Database 21.09.2015

```
    database_path: %kernel.root_dir%/data.db3
    database_user:
    database_password:
```

After this, we make changes to the app/config/config.yml file:

```
path:      "%database_path%" #uncomment this line
...
driver:    "%database_driver%" #change driver to this
```

4.2.2 MySQL

We change the app/config/parameters.yml like this:

```
parameters:
    database_driver: pdo_mysql
    database_host: localhost
    database_port:
    database_user: root
    database_password:
    database_name: database
```

After this, we make changes to the app/config/config.yml file:

```
driver:    %database_driver% #change driver to this
```

4.2.3 Initialize the database

If we try to open `http://localhost:8000/article/` we get a nice SQL error saying, that the table “article” doesn’t exist. So we create the database and the tables:

```
$ php app/console doctrine:database:create   //create the database (
    SQLite: file, MySQL: database)

Created database C:\xampp\htdocs\SymfonyTutorial\app/data.db3 for
    connection named default

$ php app/console doctrine:schema:create  //create the tables

ATTENTION: This operation should not be executed in a production
    environment.

Creating database schema...
Database schema created successfully
```

The basic application is hereby finished and working: `http://localhost:3000/article`.

4.3 Adding a date field

After now having a working application, we would like to add a new field to the Article entity.

We add a new class variable to the Article entity file src/AppBundle/Entity/Article.php. As type we choose “datetime” which is supported by Doctrine.

```
[...]
/**
 * @var date
 *
 * @ORM\Column(name="date", type="datetime", nullable=true)
 */
private $date;
[...]
```

To automatically add the getter and setter methods for this new variable, we execute the corresponding command:

```
php app/console doctrine:generate:entities AppBundle:Article
```

4.3.1 Database migration

If we now open `http://localhost:8000` a SQL error saying there is no such column named “date” is displayed. To fix this, we need to update the database. We’ll use the Doctrine migrations bundle[12] for this.
To install the bundle, we first need to install composer[13], a dependency manager for PHP:

```
$ composer require doctrine/doctrine-migrations-bundle "^1.0"
#!/usr/bin/env php
All settings correct for using Composer
Downloading...

Composer successfully installed to: C:\xampp\htdocs\SymfonyTutorial\
    composer.phar
Use it: php composer.phar
```

[12]`http://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html` 21.09.2015
[13]`https://getcomposer.org/`

After that, we install the bundle with composer:

```
$ php composer.phar require doctrine/doctrine-migrations-bundle "^1.0"

./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing doctrine/migrations (v1.0.0)
    Downloading: 100%

  - Installing doctrine/doctrine-migrations-bundle (1.0.1)
    Downloading: 100%

Writing lock file
Generating autolad files
[...]
```

Finally, we have to add the bundle to the app/AppKernel.php file by including the following code:

```
public function registerBundles()
{
    $bundles = array(
        //...
        new Doctrine\Bundle\MigrationsBundle\DoctrineMigrationsBundle
    (),
    );
}
```

We can now automatically generate migrations[14]. The migration procedure compares the Doctrine mapping information with the actual current database structure. The following command generates a migration file containing all necessary SQL commands:

```
$ php app/console doctrine:migrations:diff
Generated new migration class to "C:\xampp\htdocs\SymfonyTutorial\app
   /DoctrineMigrations/Version20150921103621.php" from schema
   differences.
```

[14]http://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html#generating-migrations-automatically 21.09.2015

Next, we run the migration to add the new entity field to the database:

```
php app/console doctrine:migrations:migrate

                    Application Migrations

WARNING! You are about to execute a database migration that could
   result in schema changes and data lost. Are you sure you wish to
   continue? (y/n)y
Migrating up to 20150921103621 from 0

  ++ migrating 20150921103621

     -> ALTER TABLE article ADD COLUMN date DATETIME DEFAULT NULL

  ++ migrated (0.08s)

  ------------------------

  ++ finished in 0.08
  ++ 1 migrations executed
  ++ 1 sql queries
```

4.3.2 Updating the form

The last important file to modify is the form type `src/AppBundle/Form/ArticleType.php`:

```php
public function buildForm(FormBuilderInterface $builder, array
   $options)
{
    $builder
        ->add('title')
        ->add('text')
        ->add('date') // add this line
    ;
}
```

This will add a special date field to the form. It adds dropdowns for day, month, year, hours and minutes.

4.3.3 Extending the templates

Only the show and index templates have to be edited.
src/AppBundle/Resources/views/Article/index.html.twig:

```
[...]
<table class="records_list">
   <thead>
      <tr>
         [...]
         <th>Date</th>
         [...]
      </tr>
   </thead>
   [...]
   {% for entity in entities %}
      <tr>
         [...]
         <td>{{ entity.date is null ? '' : entity.date|date("d.m.Y -
   H:i") }}</td>
         [...]
      </tr>
   {% endfor %}
   [...]
</table>
[...]
```

src/AppBundle/Resources/views/Article/show.html.twig:

```
[...]
<table class="record_properties">
   <tbody>
      <tr>
         [...]
         <th>Date</th>
         <td>{{ entity.date is null ? '' : entity.date|date("d.m.Y -
   H:i") }}</td>
      </tr>
   </tbody>
</table>
[...]
```

5 What's Next?

There are multiple things we could implement:

- Use a CSS framework like Bootstrap[15]
- Image uploading for blog entries
- User management system with administrators and authors
- Share entries (Facebook, Twitter, etc.)
- Third party authentication (OAuth, Google, Facebook, Twitter, etc.) e.g. with HWIOAuthBundle[16]

For many things, there are bundles available. A big database is Knp Bundles[17].

Symfony's documentation[18], The Symfony (Cook)Book, provides further information and instructions.

[15] `http://getbootstrap.com/`
[16] `https://github.com/hwi/HWIOAuthBundle`
[17] `http://knpbundles.com/`
[18] `http://symfony.com/doc/current/index.html` 6.10.2015

Appendix B: Software Requirements Specification

Contents

1 Introduction

1.1 Purpose

Make scheduling in a group of persons easier. All involved persons should be able to quickly see an overview over active polls and the deadlines or the fixed dates (or results) of closed polls. The users should be reminded of polls where they have not voted yet.

1.2 Stakeholder

Sports club "Unihockey Lohn", especially the president Roger Eichenberger and Pascal Müller.

1.3 Requirements overview

A brief overview, what functions should be implemented in the software.

General

- Mobile website app (responsive view)

Basic

- Create groups
- Different ranks in one group (with read/write access restrictions for polls)

- group overview with information (polls, deadlines, calendar)
- Schedule poll
- Text poll
- Every poll has a public link (not indexable)

Schedule poll

- Poll creator can set a deadline
- After deadline, poll creator can fix the best date
    - System shows the best possibility
- Possibility to insert first date/time in other options

Text poll

- Simple text poll
- Combine text and schedule poll

User system

- User registration
- For each poll, it can be specified, if the user has to create an account or log in
- Import users (.csv, etc)
- User can belong to several groups
- User can have several ranks in a specific group
- Admin rights for creating polls

Calendar

- Calendar with all deadlines and fixed dates
- .ics calendar service for clients (Outlook, Smartphone, etc.)

Notifications

- E-Mail
- Deadline reminder
- Mail dispatch for new polls to all target users
- Disable/Enable each type of notification

1.4 References

Which technologies will be used?

2
Use cases

2.1 List of all polls that have been opened sometime

Actors

Primary: User, Secondary: Data system

Description

As a user, I want to see a complete list with all polls, Ive ever opened. I want to be able to delete and open entries.
Also, Id like to see the deadline, start date and in case, the poll has ended, the fixed result.

Triggers

User logs in and sees a home view with the list.
Also the “Home” button leads to this view.

Pre-conditions

1. First use: User must register
2. User must be logged in

Post-conditions

1. User sees a list of all polls he’s ever opened
   a. Deadline and start date
   b. Fixed result if poll has ended
2. User can open entries
3. User can remove entries from list

Main scenario

1. User logs in or navigates to the home view
2. System loads all the relevant data
3. System shows a list of all the user’s polls

Alternative scenarios

1. User has no polls viewed yet
   a. System shows an info message

2.2 List of all groups the user is member of

Actors

Primary: User, Secondary: Data system

Description

As a user, I want to see a list of all groups Im member of with all associated polls.

Triggers

User logs in and navigates to the group overview page

Pre-conditions

1. First use: User must register
2. User must be logged in

Post-conditions

1. User sees a list of all groups and associated polls
   a. Group name
   b. Deadline and start date
   c. Fixed result if poll has ended
2. user can open entries

Main scenario

1. User logs in or goes to the group overview
2. System loads all the relevant data
3. System shows a list of all the groups and associated polls

Alternative scenarios

1. User is not a member of a specific group
   a. System shows an info message

2.3 Create a user account

Actors

Primary: User, Secondary: Data system

Description

As a user, I have to register an account in order to have access to all features.

Triggers

1. User opens a poll link that requires an account
2. User goes to the register page

Pre-conditions

1. User does not already have an account with his email address

Post-conditions

1. The form has been filled out correctly
2. The user account has been registered (in the database)
3. User gets a welcome email with a confirmation link
4. ser gets redirected to the login page (he must not first confirm the registration in order to log in)

Main scenario

1. User goes on the website and clicks register
2. The registration site is displayed
3. The user fills in the form and submits it
4. The account gets created in the system

Alternative scenarios

1. User opens a poll link that requires an account
    a. The user does not have an account
        i. A info message with registration form is displayed
    b. The user has an account
        i. Log in form is displayed

2.4 Restore the account password

Actors

Primary: User, Secondary: Data system, Tertiary: Email system

Description

As a user, I want to restore my password in case I forgot it.

Triggers

1. User fails to log in
2. User goes to the password recovery page

Pre-conditions

1. User does already an account with his email address (and knows the email address)

Post-conditions

1. The form has been filled out correctly
   a. Email address
2. User received restore email with a restore link
3. After opening the link, the user receives a new password

Main scenario

1. User goes on the website and clicks login
2. User doesnt know his password
3. User clicks on forgot password
4. The password restore site is displayed
5. The user fills in the form and submits it
6. The user receives an email with a restore link
7. after opening the restore link, the user receives a new password per email

2.5 Create a schedule poll

Actors

Primary: User, Secondary: Data system

Description

As a user, I want to create a schedule poll to find a suitable date for an event.

Triggers

1. User goes to the poll creation site
    a. The user chooses schedule poll

Pre-conditions

none

Post-conditions

1. The poll is created
2. The user gets a link for sharing the poll

Main scenario

1. User goes on the website, logs in and creates a poll
2. The poll creation form is displayed
3. The user fills in the form, possible dates and submits it
    a. Set deadline
    b. Set title
    c. Set location
    d. Set description
    e. Set all possible dates to choose
4. The poll gets created an a sharing link is displayed

Alternative scenarios

1. User wants to create a poll without an account
   a. The user cant edit the poll after creating it
2. User wants to create the poll in a specific group
   a. The user chooses the group first
3. Its a group poll
   a. Set other settings
      i. Read/write rights for specific ranks if there are any

Special requirements

1. Special JavaScript libraries for an accurate presentation of the date choosing.

2.6 Edit a schedule poll

Actors

Primary: User, Secondary: Data system

Description

As a user, I want to edit a created schedule poll.

Triggers

1. User goes to the poll directly with the sharing link
    a. The user clicks on edit
2. User goes to his account and clicks edit or goes to the poll

Pre-conditions

1. The user is logged in
2. the user must be owner of the poll
3. The poll must have been created when the user was logged in

Post-conditions

1. The poll is edited
2. All poll participants with an account got notified of the change

Main scenario

1. User goes on the website, logs in and edits a poll
2. The poll edit form is displayed
3. The user updates the settings and submits it
    a. Edit deadline
    b. Edit title
    c. Edit location

   d. Edit description
   e. Edit/add/remove possible dates to choose

4. The poll gets updated and the same sharing link is displayed

Alternative scenarios

1. Its a group poll
   a. Update other options
      i. Update read/write rights for specific ranks if there are any
   b. All administrators can edit the poll

Special requirements

1. Special JavaScript libraries for an accurate presentation of the date choosing.

2.7 Create a text poll

Actors

Primary: User, Secondary: Data system

Description

As a user, I want to create a text poll to make a simple poll.

Triggers

1. User goes to the poll creation site
    a. The user chooses text poll

Pre-conditions

none

Post-conditions

1. The poll is created
2. The user gets a link for sharing the poll

Main scenario

1. User goes on the website, logs in and creates a poll
2. The poll creation form is displayed
3. The user fills in the form, possible dates and submits it
    a. Set deadline
    b. Set title
    c. Set location
    d. Set description
    e. Set all possible options to choose
4. The poll gets created an a sharing link is displayed

Alternative scenarios

1. User wants to create a poll without an account
   a. The user cant edit the poll after creating it
2. User wants to create the poll in a specific group
   a. The user chooses the group first
3. Its a group poll
   a. Set other settings
      i. Read/write rights for specific ranks if there are any

Special requirements

1. Special JavaScript libraries for an accurate presentation of the date choosing.

2.8 Edit a text poll

Actors

Primary: User, Secondary: Data system

Description

As a user, I want to edit a text poll I have created.

Triggers

1. User goes to the poll directly with the sharing link
   a. The user clicks on edit
2. User goes to his account and clicks edit or goes to the poll

Pre-conditions

1. The user is logged in
2. the user must be owner of the poll
3. The poll must have been created when the user was logged in

Post-conditions

1. The poll is edited
2. All poll participants with an account got notified of the change

Main scenario

1. User goes on the website, logs in and edits a poll
2. The poll edit form is displayed
3. The user updates the settings and submits it
   a. Edit deadline
   b. Edit title
   c. Edit location

   d. Edit description
   e. Edit/add/remove possible dates to choose

4. The poll gets updated and the same sharing link is displayed

Alternative scenarios

1. Its a group poll
   a. Update other options
      i. Update read/write rights for specific ranks if there are any
   b. All administrators can edit the poll

Special requirements

1. Special JavaScript libraries for an accurate presentation of the date choosing.

2.9 Delete/cancel a poll

Actors

Primary: User, Secondary: Data system, Tertiary: Registered participants

Description

As a user, I want to cancel/delete a poll.

Triggers

1. User goes to the poll directly with the sharing link
   a. The user clicks on delete
2. User goes to his account and clicks delete or goes to the poll

Pre-conditions

1. The user is logged in
2. The user must be owner of the poll
3. The poll must have been created when the user was logged in

Post-conditions

1. The poll is edited
2. All poll participants with an account got notified of the change

Main scenario

1. User goes on the website, logs in and opens the poll
2. The poll overview is displayed
3. The user clicks on the “Delete” button
4. He confirms the removal of the poll.
5. The poll is cancelled and all participants get notified

2.10 Create a group

Actors

Primary: User, Secondary: Data system, Tertiary: Other users/group members

Description

As a user, I want to create a group with members

Triggers

1. User goes to the group creation site

Pre-conditions

1. The user is logged in

Post-conditions

1. The group is created
2. The user can add members to the group based on their email addresses

Main scenario

1. User goes on the website and creates a group
2. The group creation form is displayed
3. The user fills in the form and submits it
   a. Set group name
4. he group gets created

Alternative scenarios

1. User wants to create a poll without an account
   a. he user cant edit the poll after creating it

2.11 Edit a group

Actors

Primary: User, Secondary: Data system, Tertiary: Other users/group members

Description

As a user, I want to edit a group I have created or Im administrator of.

Triggers

1. User goes to the group overview site.

Pre-conditions

1. The user is logged in
2. The user is administrator of the group

Post-conditions

1. The group is edited
2. The user can add/delete members to the group based on their email addresses

Main scenario

1. User goes on the website and edits a group
2. The group edit form is displayed
3. The user updates the form and submits it
    a. Edit group name
4. The group gets updated

2.12 Delete a group

Actors

Primary: User, Secondary: Data system, Tertiary: Other users/group members

Description

As a user, I want to delete a group I have created or Im administrator of.

Triggers

1. User goes to the group overview site.

Pre-conditions

1. The user is logged in
2. The user is administrator of the group

Post-conditions

1. The group is deleted
2. All associated polls are deleted
3. All members are not member of the group anymore

Main scenario

1. User goes on the website and deletes a group
2. All associated polls gets deleted
3. All members get a notification
4. The group gets deleted

Special requirements

1. Especially ask if the user really want to delete the group, if there are members or associated polls.

2.13 Add ranks to a group

Actors

Primary: User, Secondary: Data system, Tertiary: group members

Description

As a user, I want to add ranks to a group.

Triggers

1. User goes to the group overview page

Pre-conditions

1. The user is logged in
2. The user is administrator of the group

Post-conditions

1. The specific ranks are added to the group
2. Members are assigned to specific ranks

Main scenario

1. User goes on the website and adds ranks to a group
2. The rank adding form is displayed
3. The user fills in the form and submits it
   a. Set rank names
4. The ranks are added
5. The user assigns ranks to some members

Alternative scenarios

1. User wants to create a poll without an account
    a. The user cant edit the poll after creating it

2.14 Assign ranks to group members

Actors

Primary: User, Secondary: Data system, Tertiary: group members

Description

As a group administrator, I want to assign ranks to group members

Triggers

1. User goes to the group overview page and clicks on member management

Pre-conditions

1. The user is logged in
2. The user is administrator of the group
3. There already are some ranks created in the specific group

Post-conditions

1. The specific ranks are assigned to the chosen members

Main scenario

1. User goes on the website and assigns ranks to members
2. All group members are displayed
3. The user can set read/write rights for each member/rank
4. The ranks get assigned

Appendix C: Technologies evaluation

Contents

1 PHP Framework

1.1 Overview

Available frameworks that have a good amount of features and have support:

- CakePHP
  - Not that wide and has not many functions
- Symfony2
  - Very promising
  - Has many good features
  - From SensioLabs
- Laravel
  - Looks also good, has a few features that Symfony hasnt
- Silex
  - Micro Framework
  - From Sensiolabs
  - Too small, only a few Features

The evaluation will compare Symfony2 and Laravel, because they are the two best looking frameworks actually available.

1.2 Evaluation

1.2.1 Mailing

Both frameworks use Swift Mailer. It is very accurate and provides all needed functionality.

1.2.2 Template Engine

Symfony Symfony uses Twig, a very powerful template engine with many options, filters and extensions. It includes a caching system.
Laravel It uses Blade, a template language more like PHP. Is also powerful and has many functions. Using translation in Twigs template engine is much easier than in Laravel, because Twig provides a simple filter: variable—trans .
Conclusion Twig as a template engine feels right. But Blade would also be a good choice.

1.2.3 Routing

Both frameworks provide a routing class which is easy to configure. In Symfony2 the route can be specified inside the controller. In Laravel, the routes are specified in a separate global file.
Conclusion Is not important, both frameworks implement the routing pattern fine.

1.2.4 Sessions / Security / Authentication

Symfony CSRF-Protection: possible CSRF: Cross-Site-Request-Forgery
Authentication can be based on roles, no real example included.
Session system / variables are provided.
Laravel CSRF-Protection: possible Basic authentication service and technologies already included as example. Session system / variables are provided. Also it supports authentication with OAuth Providers (Facebook, Google, etc.)
Conclusion Laravel has the better setting to start, but Symfony provides good basics for different user levels.

1.2.5 Translation

Symfony Translations are specified in YAML, XML or PHP in separate files. YAML would be the easiest syntax to use. Also, the template engine provides a simple filter to directly get the translation of a sentence. It supports fallback mode and several reading formats like simple PHP arrays, XML, YAML, gettext, Json, csv, etc. Language strings can contain placeholders for variables. The definition of one translation can be a whole

string or only keywords. E.x.: “Symfony is great” or “symfony.great”.
Laravel Laravel implements a part of Symfony‘s translation extension. It just supports PHP arrays as translation definitions and also placeholders in translation strings. Conclusion: Symfony handles translations better, because it‘s much more intuitive and powerful, if the language key is the complete sentence, also for reading the code.

1.2.6 Data Management

Symfony Symfony implements Doctrine 2.0, which is an ORM (object-relational mapping) framework with the Data Mapper pattern that supports many database management systems like MySQL, PostgreSQL; NoSQL, MongoDB, etc. It‘s a powerful database abstraction layer (via PHP PDO) and objects are created with PHP and it let the Repository Manager handle the persistence. Direct mapping of object variables to database with metadata inside the class file is also implemented. This method is more explicit, as it is needed to declare the database structure direct in the entity.
Laravel Laravel contains Eloquent, which also is an ORM framework, but it implements the Active Record pattern. It is more simple and lightweight, but doesn‘t provide so much functionality and the database structure being defined in an external global file limits the overview over an entity.
Additional Why would you want to use Doctrine2 instead of Eloquent?[1]

The benefit of using Doctrine2 over Eloquent is that the domain logic of the object is kept completely separate from the logic of persisting data to the database. This means that the things that make your application unique, don‘t need to be concerned with how the data is actually persisted to storage.

When you use an Active Record implementation like Eloquent, each object has the ability to change the database. This makes for code that is leaky and can cause issues if the rules of dealing with those objects arent strictly followed.

Using a Data Mapper implementation like Doctrine 2 ensures that the business logic of the objects are encapsulated in plain PHP objects, and all logic about how those objects are stored is handled by a completely separate service. It‘s not that easy to manage the database, because it must be created in a separate file and not in the entity model.
Conclusion So in conclusion, Doctrine seems better for handling and overview.

[1]`http://culttt.com/2014/06/30/getting-started-doctrine-2-laravel/` 15.02.2015

1.2.7 Forms

Symfony Nice implementation of forms in the template engine, it provides a simple way to render them. The form classes can be extended with constraints and validations.
Laravel There is not much about forms in the documentation of Laravel 5.0, but in 4.2 there are some basic things mentioned. Laravel handles requests different in comparison to Symfony.
Conclusion The forms have similar functions, and both would be a good choice.

1.2.8 Validation

Symfony Validations and constraints can be also defined as assertion-annotations directly in the form class. There exist constraint groups, so for one object specific constraints can be defined, depending on the usage.
Laravel Has a simple, convenient facility for validating data and retrieving validation error messages. It also supports form validation, but for this, it needs a separate class. It also supports authorizing form requests. So if a user wants to update his own post for example, it checks if its really his post.
Conclusion The Symfony validation features are pretty awesome and beat Laravel.

1.2.9 3rd party bundles

Template switching

Symfony Symfony2: A nice bundle that handles template fallback in a cool way: `https://github.com/liip/LiipThemeBundle`
Laravel There are Agent Bundles that check the User Agent, but no bundle for template fallback with User Agent checking.

1.3 Conclusion

	Symfony2	**Laravel**
User Authentication	Basic, role-based, extendable	Included, extended, no native roles/permissions
Secured areas	Yes	Yes
Security (CSRF)	Yes	Yes
Session variables	Yes	Yes, easy to handle
Template Engine	Twig: really cool and powerful	Blade: nice
Language Translator	Nice, with Twig extension	Good
Database Entity Model Management	Doctrine 2, extended	Eloquent, good
Mobile template switching/overriding	Complicated, but powerful bundle available	No good extension available
MVC	Yes	Yes
Folder structure	Good and clear	Okay
Supports extensions (bundles)	Yes	Yes
Unit testing	Yes	Yes
REST API	Not native, easy with bundle	Yes
Overall documentation	Very good	Good (Video tutorials available)
Overall complexity	High	Middle

The Symfony framework is more attractive because of many features that appear better reasoned than they do in Laravel (for me personally). The Data Mapper pattern provides a good overview over database objects, where Laravel uses separate sources.

Laravel has some nice features, like Facades. Facades provides a static object to access an object instance of core components. So its easier to access it, with only just a static reference like “DB::query()”.

Both frameworks have big communities, while Symfony2 is almost going to be an industrial standard. There are also more bundles (extensions) available.

At the end, it also matters, which framework the developers prefer. So the decision is made for Symfony2. It‘s a big and powerful framework, which will meet the requirements.

1.4 Sources

http://symfony.com/doc/current/index.html 16.02.2015
http://laravel.com/docs/5.0 16.02.2015
http://laravel.com/docs/4.2 16.02.2015
http://phpixie.com/blog/thoughts-on-using-facades/ 16.02.2015
http://de.wikipedia.org/wiki/Cross-Site-Request-Forgery 16.02.2015
https://github.com/liip/LiipThemeBundle 16.02.2015
http://swiftmailer.org/ 14.02.2015
https://www.flynsarmy.com/2015/02/creating-a-basic-todo-application-in-laravel-5-part-1/ 15.02.2015
http://tutorial.symblog.co.uk/ 14.02.2015
http://www.phpframeworks.com/ 13.02.2015
https://github.com/showcases/web-application-frameworks 13.02.2015

2 HTML front-end framework

There exist two big HTML frameworks: Twitter Bootstrap[1] and Foundation[2]. Both frameworks have good and usable features like responsiveness.
We choose Twitter Bootstrap, because there are many HTML themes that it supports and it has good components.

To get a nice looking template, we went for a ready-to-use Symfony bundle called AdminThemeBundle[3] that implements the AdminLTE[4] theme.

1 `http://getbootstrap.com/`

2 `http://foundation.zurb.com/`

3 `https://github.com/avanzu/AdminThemeBundle` 20.03.2015, Version 1.3.1

4 `https://almsaeedstudio.com/preview, Version 2.3.1` 20.03.2015