IMS - Project Documentation
UniScout Website

A6 - Giulia Baldini, Emanuela Calabi, Mikel Grabocka

January 25, 2018

For this project we have tried to work as a group as much as possible. We managed to work both singularly and together, thus all decisions that were taken reflect the whole team's point of view.

1 User Stories

The starting point for our project was deciding who we wanted to be and the role we wanted to play. We decided we could be a student association of the Free University of Bolzano named UniScout, who organises events and trips around South Tyrol, in which people can take part, with the goal of making them discover the beauty and richness of this region.
Finally, we decided to build a website for such association so that we could spread the voice about our events and gain reputation among students. After gathering ideas, we came up with the following three user stories, which summarise various scenarios that our possible users are likely to experience.

1.1 The new student

A new student of the Free University of Bolzano arrives in the city. He/she does not have any friends, and neither has he/she been in South Tyrol before. However, he/she wants to discover the beauty of the villages and the environment in which he/she will live for some years. He/she does not want to visit alone and does not have any information regarding what to see and how to get there.

1.2 The existing student

A student of Free University of Bolzano that wants to travel around South Tyrol. He/she does not want to travel alone, but he/she does not feel like putting all the effort in gathering companions and organising a trip, nor can he/she handle all the stress for this.

1.3 Not a student of Free University of Bolzano

Someone outside of the Free University of Bolzano is interested in visiting South Tyrol. The best example in this case would be a friend or relative of a student that comes in the region to visit him/her. Since UniScout organizes trips for everyone, they would be allowed to participate.

2 Implementation and Technologies

For the implementation of the UniScout website we used several technologies. In addition to the standard HTML5 and CSS we used Bootstrap for the front-end. JavaScript, AJAX, JQuery, PHP, MySQL were our choices for the back-end.
Moreover, we used the following external libraries:

- **Lightbox:** for enlarging pictures on mouse click in and navigating through them in the Home page and Past Events page;

- **PHPMailer:** to send confirmation emails to the people who sign themselves up for events in the Home page and to send reminder emails to the participants on the day before the event they are signed up for;
- **Mail Scheduler by Chintan Patel:** to schedule automatic dispatch of reminder emails every day at a specific time.

2.1 Front-end

Our goal for the front-end was to make our website as responsive as possible, so that it would display properly both on desktop and on mobile. For this reason we included Bootstrap in its development. As a result, it adjusts to all screen sizes and, thus, to different devices. One of the Bootstrap features we found to be the most useful was its grid system. In addition to this, Bootstrap's classes, with pre-defined style, and the collapsing navigation bar menu for the mobile version were also a choice.
We also decided to use the Lightbox library. This choice was due to the fact that our website contains several pictures to attract the users. In this context, having the possibility to enlarge the pictures to full screen is not mandatory, but definitely nice to have so that the user can visualise what the event is about. The forementioned library was used in the Home and Gallery pages, where the photos of both upcoming and past events are shown.

2.2 Back-end

Our choice of the back-end technology was made accordingly to the tools we learned during the course Internet and Mobile Services. The core functionalities of the back-end implement all the technologies we mentioned above.
More in detail, the procedure for loading events in the Home page works as follows:

1. Firstly, we use AJAX to call a PHP file (Figure 1);
2. Secondly, in the PHP file we execute a query and we return a JSON-format result (Figure 2);
3. Thirdly this result is collected with JavaScript and processed with JQuery (Figure 3).

As you can see we create a div element for each post, then we insert it into the div element with id="posts" in the `home.html` file. The template for each post is retrieved from the `home.html` file, and we modify it in this part.
To save a new participant for an event in our database, we retrieve the the post id. In fact we set the template to contain the post title and id. The latter one is saved as a hidden field in the form. We need to do so, as we will need it to match the participant id with the event id when querying the database in the saving process.
We also created two attributes for each post, namely shortText and fullText. Short text is a reduced version of the full text, the first 250 characters at the moment, so that long posts do not take up too much space and the user does not need to scroll down excessively.
Upon clicking the "Read more" button at the bottom of the post, the shortText is switched in favour of the fullText and the user can read the full event description. The button is switched to display the "Hide" command which reverses the action of the former, as you can read in Figure 4.
We have another div element called container, inside the post, which contains the form that the user must fill in order to participate. As above, this form is showed only when pressing the button "Read more".

The external libraries for the back end, namely PHPMailer and the Mail Scheduler, are used for sending emails to the users when they register for an event and to schedule an email that is sent the day before the event as a reminder for the participants.

```
function getPosts() {
        if (window.XMLHttpRequest) {
                xmlhttp = new XMLHttpRequest();
        } else {
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        var result;
        xmlhttp.open("GET", "php/load_posts_home.php", false);
        xmlhttp.onreadystatechange = function() {
                if (this.readyState == 4 && this.status == 200) {
                        result = this.responseText;
                }
        };
        xmlhttp.send(null);
        return result;
}
```

Figure 1: AJAX call

```
$sql = "SELECT id_event, title, date_time, location, description,
photo FROM events WHERE date_time > CURRENT_TIME()
ORDER BY date_time ASC;";
$result = $conn->query($sql);
$rows = array();
while($row = $result->fetch_assoc()) {
        $rows[] = $row;
}
print json_encode($rows);
$conn->close();
```

Figure 2: PHP code

```
var jsonPosts = JSON.parse(getPosts());
for (var i = 0; i < jsonPosts.length; ++i) {
        var postId = "post" + i;
        $("#posts").append("<div id='" + postId + "'></div>");
        var post = $("#"+postId);
        $("#postTemplate").children().clone().appendTo(post);
        $(post).find(".postTitle").text(jsonPosts[i].title);
        var content = jsonPosts[i].description;
        var partial = content.substring(0,250) + "...";
        $(post).find(".postContent").text(partial);
        $(post).data("id", jsonPosts[i].id_event);
        $(post).find(".postContent").data("shortText", partial);
        $(post).find(".postContent").data("fullText", content);
        $(post).find(".postImage").attr("src",jsonPosts[i].photo);
        ...
        $(post).find(".id").attr("value",jsonPosts[i].id_event);
        $(post).find(".image-link").attr('href', jsonPosts[i].photo);
        $(post).find(".image-link").attr('data-lightbox', postId);
        $(post).find(".image-link").attr('data-title', jsonPosts[i].title);
}
```

Figure 3: JQuery processing

```
$(".btnRead").click(function(){
        var button = $(this);
        var postContent = $(button).parent().siblings().children(".postContent");
        var container = $(button).parent().siblings(".row").children(".container");
        $(container).toggle();
        if($(container).css('display')=='none'){
                $(postContent).text($(postContent).data("shortText"));
                $(button).html("Read more");
        }else{
                $(postContent).text($(postContent).data("fullText"));
                $(button).html("Hide");
        }
});
```

Figure 4: Show more/Hide button

3 Development Process

We can divide our development process in several milestones. Firstly, we tried to create a database representing our design choice. Then, we set up the connection between the server and the database through PHP. After that, we sketched the wireframes of our website. Having those, we managed to create the website in its main parts. When we had an idea of what the actions we wanted in our pages to, we created the client-server interaction to retrieve the information from the database. At this point we achieved our least viable product. At a later stage, we started adding more features: a gallery to see all the images, a scheduler for the emails and we polished more the front-end. Next, we started testing all the features of our system. When the website was ready, we made our friends test it and we collected feedback to improve it. Lastly, another round of testing was done to check if corrections were effective.

3.1 The Database

The creation of the database was a long process: we first added the tables that we knew we needed for sure. However, it required several changes to reflect the changes in our website. The final database is composed of four tables: events, participants, articles and album.
The events table contains all the information about an event: id, title, date and time, location, description and a link to a photo describing the place.
The articles table is where the posts written about past events are saved. We imagine that in our association the admins will need to write a nice comment and add some pictures about the event that has just passed; so that they can show to possible future participants how an event works and why they should join. The articles table has the following fields: id, title, id_event (id of the event to which it corresponds), date and time of its publication and description.
The participants table holds information about the people who decided to join us on a certain event. It has contact information about the user (name, surname, phone number and email), an id and the id of the event it refers.
The album table was the added at last, since we needed more room to store all the images that were going to be in the gallery. Each image in album has its own id, the id of the event it refers, a title and link to the photo.

3.2 PHP and MySQL

After the first version of the database was ready, we immediately created a connection to PHP. We started querying the posts in order to retrieve information about the events.

3.3 Wireframes

To understand the functionalities and the features we wanted our website to have, we tried to sketch some wireframes of our final product. We agreed on a modern design, with large images to attract users with the beauty of South Tyrol.

3.4 First Steps towards the Front-End

We started creating the Home page. The first step was obviously creating the layout of the page and then the template for the posts. Secondly, the layout for About and Contact pages were created.
Lastly, we created our logo (original from Dolomiti del Brenta Trail) and chose the main colours for the website.

3.5 Connection Front-End - Back-End

At this point in time the front-end was finished. Therefore, we started thinking about the interaction with the database. We already had some PHP files ready for that, and the first approach was to create the posts server-side.
Despite the fact that it was working, we thought it would be a bit too much effort for the server, and that it would have been a better idea to have it client-side. The result is what we have now: whenever a page has some posts (Home and Past Events), we have a hidden <div> on the page, which contains our template. This template is then cloned and changed according to the event information through an AJAX call to a PHP file and JQuery, as explained above.
After all of this, we had our last viable product. The website was completely responsive, but it was only a static website showing information, and no interactions were allowed.

3.6 The New Features

Gallery Since our website is very much image-based, a gallery page gathering all the photos would be a good idea. Initially added the images, along with the caption, on that page. Afterwards, we started noticing problems with the page layout. As a solution we added a CSS-based layout which changes how many columns of images are visible depending on the width of the screen.
The page was ready, but we were concerned about the small size of the pictures in the desktop version. Therefore, we used the Lightbox library to have click-to-expand images.

Emails As a website for an association organizing trips, the need of sending email to our participants is quite realistic. We used the PHPMailer library to set up and prepare the HTML for our mails. Thanks to this, whenever someone signs up for an event, he/she will receive a confirmation message containing all the necessary information for that trip.

The Captcha Since we had the possibility to send emails to our address, we agreed that a spam control mechanism was needed. We used reCaptcha and the Google API to incorporate it on our system.

The Scheduler Another nice-to-have feature is that users receive emails from us: a week before the trip, a day before the trip, and perhaps a newsletter informing the user avout upcoming trips. In this version of our website, we implemented a reminder email on the day before the event. However, the process would be exactly the same, if we wanted to implement the other two.
Our problem with the scheduler is that we needed it to be cross-platform scheduler. This problem arised within the team due to the fact that two of us are working on a Windows machine, while the third is using Linux. For all we found out, there was no way to achieve this. The idea was then to have a Java scheduler. This solution worked for us, as long as the file keeps running. However, we are aware that this problem would not arise in a real-life scenario, as a server runs uniquely on one machine.

4 Testing

The main focus of our testing was the responsiveness of the design, because we wanted our website to be usable and look nice in any environment. Therefore, we tested all the functions of the back-end first and made sure they would yield the correct result. Then we focused on the front-end. After intense testing, we can say that we achieved our goal. In fact our website works well with any screen size and on all the main web browsers such as Google Chrome, Mozilla Firefox, Safari, Microsoft Edge and even on Internet Explorer.

To add value to the final product, we did some user-testing. Thus, we collected feedback from our colleagues on the most significant features and to refined them according to their preferences. After some adjustments, we did more rounds of testing. The result is a stable website, not affected by common issues, such as SQL Injection. In fact, we took a few step towards the goal of making it more secure.

5 Future Work

Our work could be improved by adding the following features:

- **Calendar**: a page with a simple calendar view, to better visualise when the next events are.
- **Grouping posts by tags or location**: this could be applied to the home page. The idea is be that the user can choose a tag/location from the given ones and only read those post containing such tag or that take place in such location.
- **A page for each event**: when clicking on a post in the Home or Past Events page, the user lands on a page specific for such event. This way, we could have longer pages with more information.
- **Search function**: because we organise many events, it would be easier for the user to be able to search for a specific event.