myVerein

Server Specification

Student Research Project

for the

Bachelor of Science

at Course of Studies Applied Computer Science
at the Cooperative State University Stuttgart

by

Frank Steiler

September 2014

Time of Project	13 Weeks
Student ID, Course	8216767, TINF12A
Company	Hewlett-Packard GmbH, Böblingen
Internship Supervisor	Alfred Becker

Contents

Chapter 1

Document purpose

This document is extending the general Software Requirement Specification for the whole system, specifying details about the server application. This document therefore tracks the complete design of the backend of the system, like the admin panel, the database and every other component needed to run the application.

Chapter 2

Product purpose

Chapter 3

Product requirements

Chapter 4

License

Chapter 5

Database

The server is going to use a database to store the persistent data, defined in the Software Requirement Specification. By choosing this common approach to store data, it is ensured that all information are stored consistent and persistent, even after a crash of the application.

The use of an industry standard relational database would ensure an efficient operation of the application. As a reliant and free database application, the open-source project *MySQL* is chosen.

5.1 Database tables

The following section is describing the design of the tables of the database. This is done by providing a detailed description about the functionality of each table.

5.1.1 User

All user information are stored within two tables, to ensure an extensible way to store information about the person. The *User* table is storing all required information, like the user's email or his hashed password. The *Additional Information* table is storing an indefinite amount of additional properties for each user as a key value pair attached to the user's ID.

On top of that each user is part of one ore more division. This relation is handled in a *UserDivision* table.

5.1.2 Division

Each division is an entry within the division's table. It is defined by its name and short description.

Divisions are organised hierarchical within a club, so this structure has to be represented within the database. An efficient and easy to implemented solution is needed. Since the implementation is not relying solely on SQL, the solution is not limited to *Nested Sets* or the *Materialised Path.* An *Adjacency list* seems to be a prefect fit, because its implementation is very efficient and easy to realise, but needs application side logic performing certain tasks. [Hil14]

Every user can be part of one or more divisions. This relation is defined within the *is part of* table, containing the User ID of the members and the Division ID of the divisions, as well as the date, when the user joined this division. This attribute is needed to ensure, that the application is loading all information about a new division after the user joined.

On top of that each division is administrated by a single user, who gains access to the administrator panel through this position. If there is no administrator specified, the super admin takes his role. This user is defined outside of the database, to ensure access to the panel, even if the database connection is not working.

`http://explainextended.com/2009/03/17/hierarchical-queries-in-mysql/`
`http://stackoverflow.com/questions/4048151/what-are-the-options-for-storing-hierarchi`

5.1.3 Messages

Each member of a division automatically joins a group chat between all members of this division. To ensure privacy for each chat, messages are only saved on the server as long as necessary. Therefore a *Message stack* table is created. Every sent message is stored within this table until the recipient access it, or the system deletes the message after its expiration.

Each entry contains the message to one member of the group chat. When the user syncs his application with the server, the server returns all messages from the stack for the user. The rows are deleted as soon as the user receives the message.

5.1.4 Events

A core feature of the application is creation and management of events for each division. The events are managed within the *Event* table. A event invites whole divisions, and every member can send a response to the invitation. The invited divisions are stored within the *invited* table and the responses of the users are stored within the *answered* table, which is also storing the type of answer (accepted, declined or tentative).

Besides that the event has several properties, e.g. a short description, a location, the date of the event and its last change.

5.1.5 Pictures

Since the user is able to upload pictures that are relevant to the club, a table is created to manage these pictures. The picture's metadata is handled through that table, as well as the URL pointing to the file.

Each picture is uploaded by a specific user and the user can associate up to one division to the picture.

5.2 Conceptual schema: Entity Relationship Diagram

To create a durable database it is important to have a precise plan of the design. The first step –the conceptual schema of a relational database– can be expressed as an entity-relationship model (ER-model). The specific ER-diagram for *myVerein* can be found in appendix A on page vi.

The diagram shows the standardised representation of all tables described in the previous section.

5.3 Relational schema according to Kemper/Eichler

Chapter 6

Initialisation

Which steps are needed at the first setup of the application

Chapter 7

Functionalities

List of Figures

Listings

Bibliography

[Hil14] Mike Hillyer. *Managing hierarchical data in MySQL*. 2014. URL: `http://mikehillyer.com/articles/managing-hierarchical-data-in-mysql/` (visited on 2014).

Appendices

Entity Relationship Diagram

myVerein

Author: Frank Steiler

Version: 1.0