Technical Documentation “Mediatheque”

15.10.2021

Felix BOHME

Overview

This Technical Documentation outlines the initial thoughts and planning steps of this Web Application.

A technical documentation in .pdf format with your initial thoughts on the project (technologies chosen, configuration of the work environment, class diagram or Physical Data Model, etc.) or Physical Data Model, use case diagrams, sequence diagrams, etc.).

It will be necessary to include a list of the good security practices implemented on this application, with the justification for each of them.

Goal

The given assignment for this project is to create a web application which concentrates around the main functionalities of a library. A number of users need to be given access to a catalogue of items, which the user can reserve online and pick up offline. All interactions regarding the issuing of an item from the catalogue must be recorded and put in relation to the involved entities.

Therefore two groups of users are interesting. The user himself, as well as the staff managing the user base, as well as the catalogue. Both user groups have got to have different rights and actions within the system.

I decided to bring those two user groups together in one interface, which is able to distinguish between a **user** and an **employee** and direct him to the corresponding interface.

A system of different stages for the actual process of borrowing such an item is created.

The first stage would be the presentation of a quantity of entities to a user, out of which he can choose to apply an action. In this case the initial action would be to reserve a selected entity. The user is free to repeat this process as often as he wishes.

Subsequently these reservations are being recorded and presented to the second user group which is entitled to view this information.

The action (reservation) on an item can only be performed by a user as long as the item hasn't changed it’s availability. A change of mind from the user regarding his wish to perform the action is to be considered.

Afterwards the entitled user can confirm that the intended action of the user has been executed.

After a predefined time window the user is obliged to revert his action on the item, which again has to be confirmed by an entitled user and recorded.

Specifications

Server side (local)

- Windows 10
- XAMPP Package
    - MariaDB 10+
    - PHP 7.2+
        - Ext intl extension
    - Apache 2.4+
        - With .htaccess for further heroku deployment

I chose to go with a windows installation for the server environment, because I was familiar with the XAMPP package already. Since I am regularly working in a Linux environment as well, I could have chosen this for the development. But my current office setup was not behaving as wanted with multiple screens under linux. Luckily the PhpStorm IDE is available for linux as well. That’s why I will decide to code in linux for further projects.

Front End

- HTML 5
- CSS/SCSS
- Twitter Bootstrap

Back end

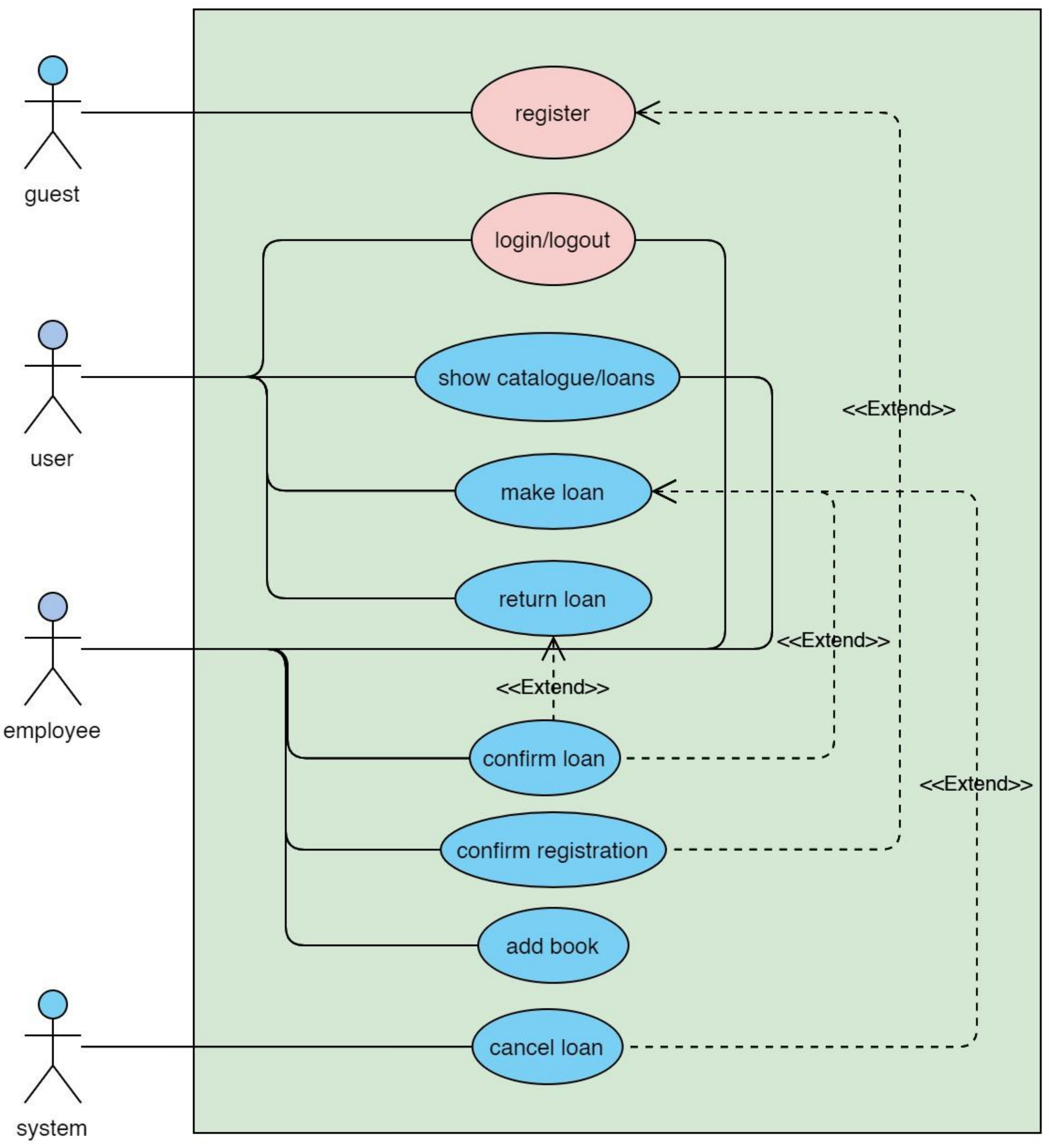
- PHP 7.2+
- Symfony with composer
- Node package manager
- EasyAdmin Bundle
- Fixtures with Faker Bundle
- MariaDB
- Git
- Heroku
    - ClearDB MySQL addon

Symfony in combination with Bootstrap is a very powerful team for any kind of coding project, in my opinion. For sure another attempt with a framework like CakePHP would have been enough, but I like the long list of possibilities with symfony.

Design

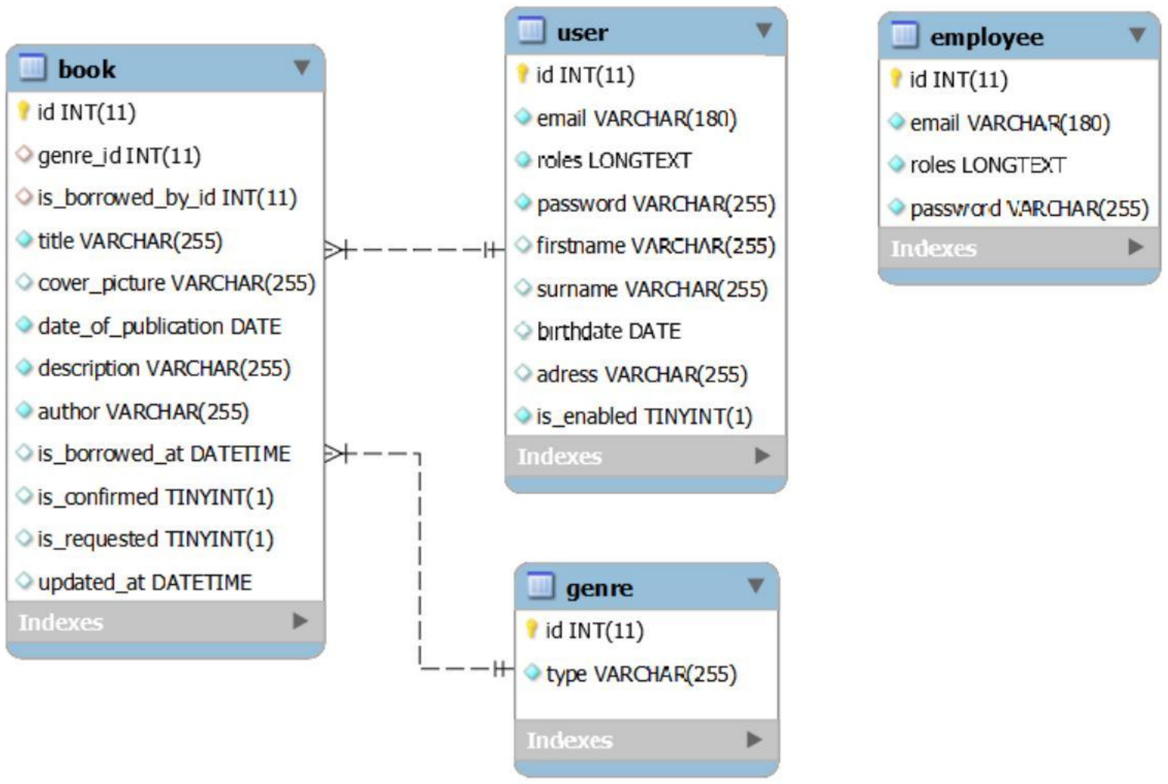
Use Case Diagram

Visual Paradigm Online Free Edition



Visual Paradigm Online Free Edition

Class Diagram



Security measures

In order to reserve the developed functions to the corresponding users, different actions have to be enrolled and activated.

The security can be divided into 2 segments:

- User Authentication
- User Authorization

To achieve secure User Authentication (Identifying and Logging in the user) the following steps have been taken care of:

- Custom User Checker Interface added in firewall
- Hashed passwords as a standard, alway use best available hasher
- CSRF Tokens in login form (Cross-site request forgery)

On the side of a secure User Authorization (giving permissions to the user) the following measures have been applied:

- Roles checked by very first Controller (HomeController)
- DashboardController checks user roles again
- Actions in Controllers are required to check permissions (roles)
- Every other route not known is not to be accessed
- Added Entities of Class User and Class Employee as User Providers to be able to login from different user groups (entities)
    - Chain User Provider
- Additional Logout Button