



# Project Work at VIA University College Horsens



# Project Report

## Visual Guide – Treasure Map

<b>Authors:</b>	Dietmar Sach, Michael Müller, Florian Wasielewski, Patrick Vogt
<b>Supervisor:</b>	Ole Ildsgaard Hougaard
<b>Project:</b>	Visual Guide – Treasure Map
<b>Date:</b>	28.09.12



## Table of contents

0. Executive Summary .....	4
0.1 Purpose.....	4
0.2 Analyses .....	4
0.3 Results.....	4
0.4 Recommendations .....	5
1. Introduction .....	6
1.1 Introductory Notes.....	6
1.1.1 Background description.....	6
1.1.2 Specification of purpose.....	6
1.1.3 Problem formulation .....	6
1.1.4 Delimitation .....	6
1.1.5 Choice of model and method procedure.....	7
2. Analysis .....	7
2.1 Library research.....	7
2.2 System research.....	7
2.3 Database model .....	8
2.4 Use Case diagram .....	9
3. Design.....	10
3.1 Look and feel .....	11
3.2 System architecture model .....	12
4. Implementation.....	13
4.1 Class diagram.....	13
4.2 The purpose of the classes.....	13
4.3 The program's structure.....	15
4.4 Functions .....	15
4.4.1 Secure administrator login.....	15
4.4.2 Mock-Up.....	15
4.4.3 Visual Guide.....	16
4.4.4 Process of constructing the map .....	16
4.4.5 Process of generating a QR-code.....	16
4.4.6 Admin-Backend.....	17
4.4.6.1 Login Process.....	17



4.4.6.2 Process of choosing a room .....	17
4.4.6.3 Process of adding new furniture .....	17
4.4.6.4 Process of deleting furniture .....	17
4.4.6.5 Process of moving furniture .....	18
4.4.6.6 Process of resetting a position .....	18
4.4.6.7 Process of changing categories .....	18
4.4.7 Operating principles of jQuery / AJAX .....	18
4.5 URL scheme .....	19
5 Testing .....	21
5.1 Documentation of the PHP classes .....	23
6 Conclusions .....	23
7 Sources, references and literature .....	24
8 Appendices .....	25



## 0. Executive Summary

### 0.1 Purpose

The purpose of this project is to make it easy for customers visiting the library of the VIA UC Horsens to find the books they are looking for. This shall be achieved by providing a visual guide, for instance a "treasure map". The visitor should be able to retrieve the visual guide after having looked-up the book on the search facilities provided by the library search system. The treasure map should be a map of the library with X'es marking where the books are located.

The librarians working at the library must be able to easily "mark the X'es", i.e. somehow provide the location of the book as part of the process of putting the book on the shelves. This process should not add significant overhead of the time it takes to shelve books.

This system should be integrated with existing library systems used by customers of the library at Campus Horsens.

Additionally, there should be a mobile solution for smartphone users.

### 0.2 Analyses

At first it is important to determine which programming language is used in the existing library system. This decides what programming language(s) will be used for the project and what development platform should be used.

A testing method for software tests must be chosen.

A decision about the map view had to be taken, the construction of the single elements on the map and how all the rooms are best realized as a map. Which rooms are relevant?

What kind of book storage system does the library have? Is it consistent and thus processable by a program? Is it changeable respectively extendable?

It is important to know exactly how the current library software works before accessing or modifying it in any way. To modify the system, access to the source code respectively an API and to the database is a precondition.

To retrieve information about the technical backgrounds and to get access to both the system and the database a contact person at the relevant positions is needed.

To not to interrupt the usual operation of the library software, best would be to work with a mock-up system or test data in a separate workspace.

For the mobile solution the user should be able to move the visual guide to a smartphone to find the way on the go.

### 0.3 Results

PHP was chosen as main programming language in the project because the existing library software apparently was built with PHP. SQL was the database of choice when using PHP. For the graphical surface the JavaScript-based framework jQuery will be used. The testing method will be UnitTests.

The existing library system is hosted by DBC, a public company owned by the Danish government, who will unfortunately not provide any access to their software nor the database



behind. Thus the existing library system cannot be modified in any way nor be supplemented with a kind of plugin.

In early 2013 a new system will be introduced to the library which is open for any modifications as the source code is publicly available. This OpenSource software is not developed by DBC but still relies on several services provided by DBC.

At the university there are contact persons at the library who provided several information about the library's structure and the scope of the project. There is no contact person that could have provided information about the technical system. The librarians made some requests to DBC to get some information.

The category system to store books is consistent in the main room but not in the other three rooms. For the other rooms there is no processable data that specifies the assignment of books to shelves or categories.

Consequently the compromise was made to focus on the main room and exclude the other rooms from the scope of our project.

To test the project software a testing system is needed. The future library software can be made operational but it does not provide any search results as the services it relies on are ought to be provided by DBC who denies giving such access for testing purposes. Thus it was decided to build a mock-up system filled with mock-up data to show exemplary usage.

The mock-up system as well as the visual guide will have its own MySQL database.

To make it possible to include the visual guide in the new library software in 2013, a sample interface will be provided.

Concerning the practical realization it was decided to display a link in the search result that opens the visual guide respectively a QR-code in a pop-up window.

The mobile solution will likely be a smartphone-optimized frontend of the book search interface and another link to a generated QR-code that opens, being scanned, the visual guide inside the mobile web browser.

Optionally, below the visual map a textual description of the book's location could be shown. Also optionally, a pop-up containing textual location information could be opened when the "X" on the map is selected.

As there won't be ever a direct access to the database, the assignment of categories to the shelves must be done manually. For the library staff there will be an admin interface to change the assignment of categories to shelves and to change the location of shelves and furniture.

## **0.4 Recommendations**

After integrating the visual guide into the productive new library software the categories have to be assigned to the shelves via the admin interface.

To be able to integrate and to use the project software, it will likely be a precondition that the new library software is installed on a local server at the university with full access to the source code.

The university is recommended to have a contact person for technical purposes who is able to maintain the library software.

It could be a recommendation to DBC to take the project as a basis to develop visual maps for other libraries, too.



## **1. Introduction**

### **1.1 Introductory Notes**

#### **1.1.1 Background description**

With the current library system of the university it is not easy for the students to determine the location of a book:

The current library web-application does not clearly display the location of a media.

Due to unclear naming of links and unnecessary additional steps it is not intuitive but very confusing for students to determine the location of a book.

Still, having the location of a book as room and category number the student still has to find the book physically on his own.

#### **1.1.2 Specification of purpose**

In the project an existing respectively new library system should be modified to ease the process of looking up books in the library for students.

The emphasis is on creating a visual guide that marks the looked up books on a map of the library.

#### **1.1.3 Problem formulation**

It is not clear how the system to develop against is being accessed as the system is not fully functional. Thus it is not possible to create a precise interface to the Visual Guide but only a rough approximation, which has to be adapted for the productive system first.

The Visual Guide required having a map section where the map is drawn consisting of ground plan and furnitures. There also should have been an admin interface where the librarians were able to change anything about the furnitures respectively shelves and categories.

The pure usage of PHP for drawing the map would generate an inflexible map where interaction is hardly possible.

To create a scheme for the database a database modeling process should be used to create the relations.

All PHP Unit-Testing software from third-party suppliers failed to work for our purposes.

#### **1.1.4 Delimitation**

The delimitation was chosen that the modifications of the old respectively the new system should be as minimal as possible. This means only the most urgent mistakes would be fixed in the programming of the old system. It also means that just a link to the map was displayed in the search result and leave as much of the program as possible unchanged.



### **1.1.5 Choice of model and method procedure**

PHP was chosen as programming language, JavaScript / jQuery 1.8 for visualization and SCRUM as software development method.

An overview of the used software and tools can be found in appendix on page 6.

## **2. Analysis**

### **2.1 Library research**

At the first visit of the library the librarians explained the current problem by reference to the library's web application. This way the issues could be experienced practically in the main room D203. The English translation of the user interface was insufficient as the location of a media was not translated and still in Danish ("Opstilling"). This value just represents the category number but not the physical location. To display the corresponding room number, a link named "more" must be used which redirects to another page where some additional steps have to be followed. This is not intuitive but very confusing to the user. Having both the room number and the category it is still not clear where the book is stored physically.

The room D201 and D202 were shown which contained magazines only and some few old books. Those rooms do not have a consistent storage system that could be processed by software. The magazines are spatial separated into the current year's issue and the other year's issues having separate shelves.

The room X4202 was not shown at the first visit of the library but later during the project. This room does not have a processible book storage system either.

Thenceforward there was a frequent correspondence with the librarians who steadily provided the latest information about the library system. Also a ground plan of the library and an database extract from DBC was provided.

### **2.2 System research**

In the beginning the librarians had to gather information about the current library system themselves. It appeared that the current system was named "integrabib" and was provided by a third-party supplier called DBC on another server.

Requests to DBC regarding access to the source code or database access ended negative so the current system could not be modified or accessed in any way.

The librarians explained that the university would introduce a new library system in 2013.

Investigations showed that the new system was probably called "Ding.TING" and was developed by the "Ting Concept Group" on whose homepage it can be downloaded.

A graphical overview of the architecture of the "Ding.TING" can be found in the appendix on page 4.





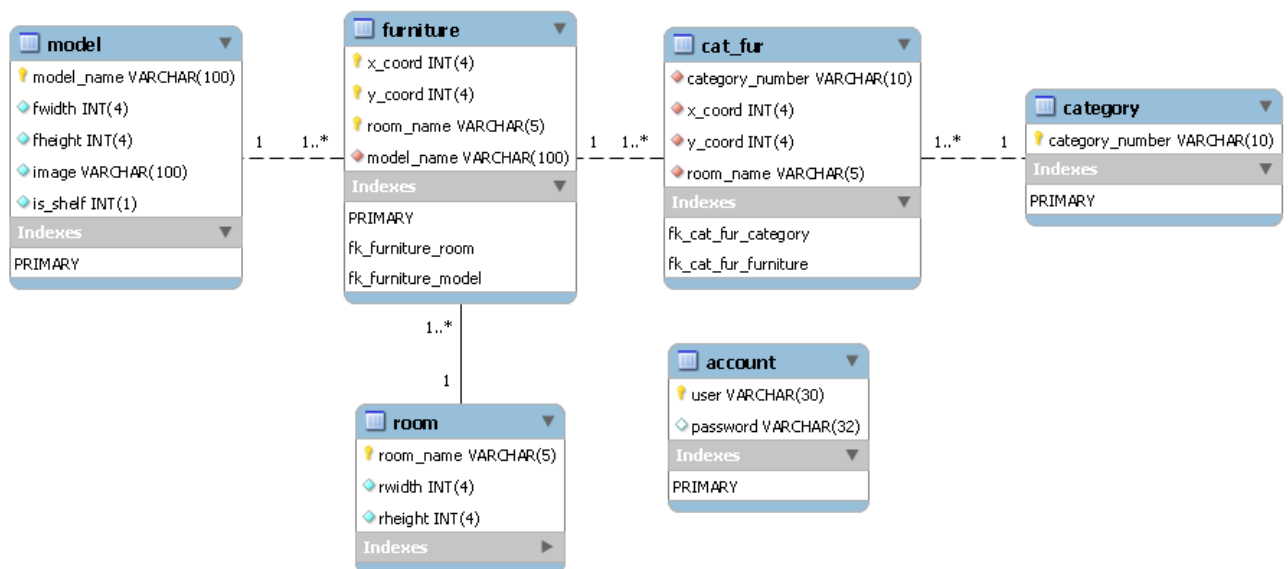
Further investigations showed that there is a successor of “Ding.Ting” that is called “ding2/Artesis” which is not developed by the “Ting Concept Group” anymore but became available Open Source.

An attempt to create an installation of the “ding2/Artesis” system for testing purpose took much time as it worked erroneous. It turned out that the working “ding2/Artesis” system relied on several services like “OpenSearch” provided by DBC which DBC denied to provide for testing usage. Thus no search results could be generated where an interface could have been implemented against.

The system “ding2/Artesis” bases on the content management system “Drupal 7” and requires several particular preconditions before an installation is possible. A description of the installation process can be found in the Appendix on page 7.

DBC agreed to provide a data extraction of their database with information about the media stored in the University Horsens. This data was used to create a mock-up system that is not a part of the project description but necessary to demonstrate the Visual Map’s functionality.

## 2.3 Database model



To get the database structure the system requirements were collected textually and semantic irreducible sentences were created (refers to appendix, page 2). From these sentences the object- and association structure of the database was created. On this basis the connected component graph was modeled and the key candidate relations were marked. From this the database model was created and formed into SQL statements (refers to appendix, page 3).

Furniture is identified by both the x/y coordinates and the room name where it is placed in. Further attributes as there are width, height (due to top view), image filename and shelf identifier flag for distinction of shelves to other furniture were outsourced into a table called “model”. A foreign key association using the attribute “model\_name” was created between “furniture” and “model”.



To save additional information like width and height of a room, another table “room” holds an identifier “room\_name” that is being referred to by “room\_name” in a piece of furniture’s record.

The table “category” only consists of a row containing a category number.

To be able to assign several categories to one shelf and to use one category in several shelves, an auxiliary table named “cat\_fur” was introduced. It makes it possible to connect both “category” and “furniture” in an “n:m”-relation.

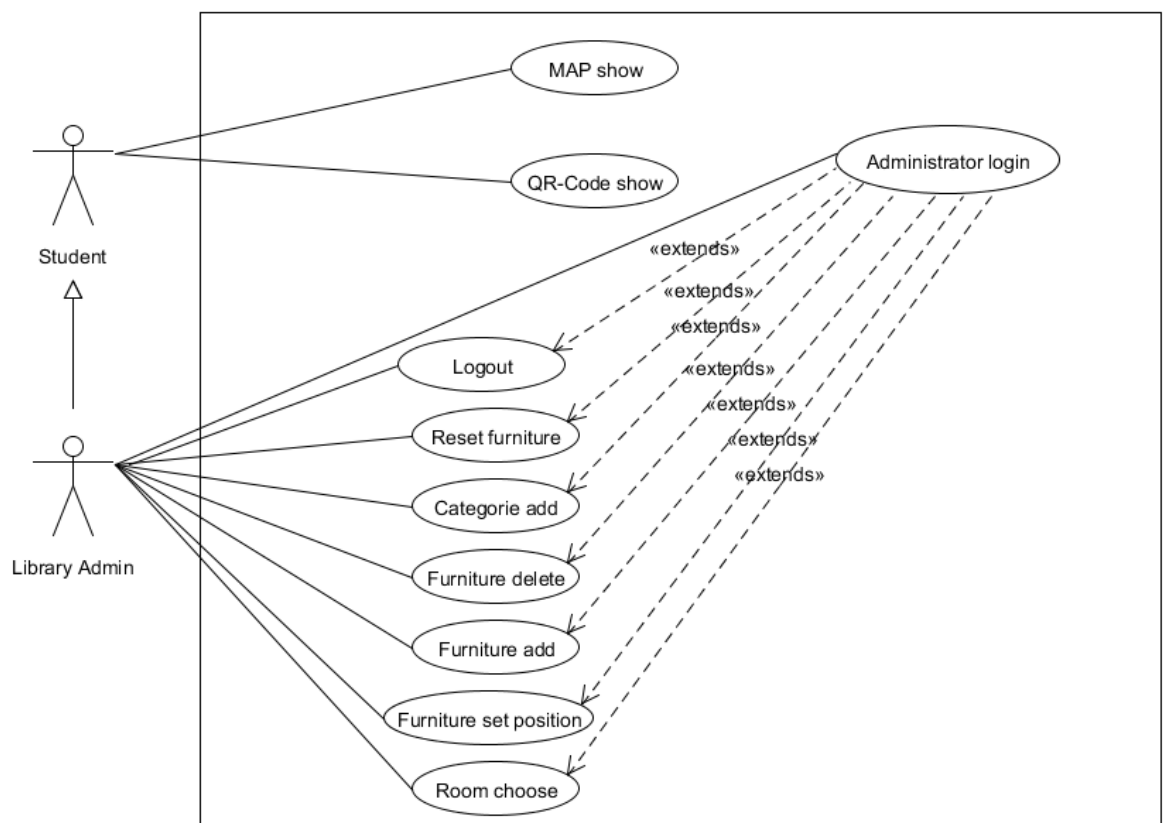
The attributes of “cat\_fur” contain a category number as well as furniture’s coordinates and room.

In “cat\_fur” all the attributes together are the primary key to prevent duplicate entries.

For the administrator login another table “account” is needed to store login data.

The SQL statements to create a database structure and to fill it with sample data are located in the project’s directory “SQL”. In this location there’s also a file with the create statements for the mock-up-page.

## 2.4 Use Case diagram



Explanation:

A student can get the position of a book displayed on the map after a book search when clicking on the corresponding link in the search results



A student has also the possibility to open a generated QR-code on click on another link and to scan the code with an smartphone app. After that on the smartphone the link to the map is being opened in a web browser.

The administrator has a wide range of functions to rely on to do several actions. As the administrator is a specialization of the student, he has all the possibilities of a student, too.

The different possibilities of the administrator contain all the functions that are available in the administrator backend like the session secured login, the selection of a room from the room list, the adding and moving of pieces of furniture or the adding and deleting of category number assignments.

The administrator is also able to do a furniture reset, which reverts all the positions to their initial position. After finishing all operations, the admin can properly log out and leave the backend using the logout button.

### 3. Design

As project should be developed with an appealing visual user interface and should be as dynamic and flexible as possible, jQuery was considered to be a good choice.

jQuery is a JavaScript framework that consists of prefabricated functions and features, which make programming a lot easier. Furthermore jQuery/JavaScript is supported by all current web browsers.

jQuery made it easier to implement the Visual Map web application and to add new features. With jQuery there was a way to create the project interactively while having as little effort as possible. jQuery is a framework of comprehensive accumulation of JavaScript methods.

jQuery offers a wide range of predefined, sophisticated functions to animate or to bind events to HTML elements. It is possible to modify CSS styles on runtime dynamically.

For instance, a great function which much advantage has been taken from is the AJAX functionality which is basically a wrapper for a predefined set of XMLHttpRequest methods. Another example is the function “live” which lets show up a category window when a piece of furniture is clicked. It will apply this event to all furniture HTML elements on whole runtime, no matter when they are added.

An important function was the jQuery function “\$(document).ready()” which prevented running any commands before the whole document was loaded.

The function “\$(this).data(‘shelf’)” reads the custom data attribute called ‘shelf’ of the selected element, for instance an <section> tag containing a shelf.

Considering technical requirements it was necessary to use a programming language like PHP which supports database access. The programming language PHP is also a good choice when developing web applications. PHP is supported by virtually every server. PHP was used to process the JSON-encoded calls sent by JavaScript. Additionally it was found to be important to use an object-oriented language, so we could use numerous object-oriented elements, for example interfaces or inheritance of classes.



For the HTML-based layout of the Visual Guide map HTML5 was used, as it offers numerous new attributes and functions. It includes the latest CSS3 standard which offers great possibilities to design a website.

To name some HTML5 features used in the project:

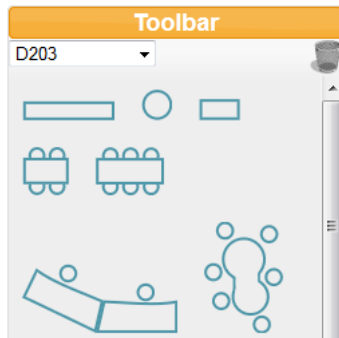
Custom data attributes were used to append metadata to append data to the HTML tags of the furniture elements on runtime. For instance, this was used to cache the furniture's position values when moving them on the map. Other data stored in data attributes of HTML elements were the model name, the image width and height and the is\_shelf flag.

Placeholder elements were used in text fields which display a default text before something is entered. It is user-friendly to display an initial value like "enter category here..." in a text box. Before HTML5 it required some effort to implement the same functionality with JavaScript. Furthermore we defined the header using the HTML5 tag "<header>" to identify the header sections more clearly.

For the mobile solution the jQuery framework was extended with the jQuery mobile framework for mobile browsers.

### 3.1 Look and feel

In the admin backend the toolbar containing furniture models is placed on the left hand side to be most intuitive. For the same reason the rectangular design elements are designed with smooth rounded edges.











The colors should have been bright, user-friendly and appealing to students while the furniture is put into contrast with the room. For instance, the blue pieces of furniture are just visualized for purposes of orientation, they cannot hold categories. Shelves which can contain categories are colored orange. The background of the room is grey as it is more appealing to the viewer's eye than white which would cause a too high contrast.



The border lines of the rooms are colored black to clearly show the room delimitation. For a better orientation the entrance is colored green. Neighbor sections like the staircase are colored light grey to clarify that they are not a part of the room itself but still on the map for orientation purpose.

Furthermore we decided the font should be Arial, as it is without serifs and available on all operating system.

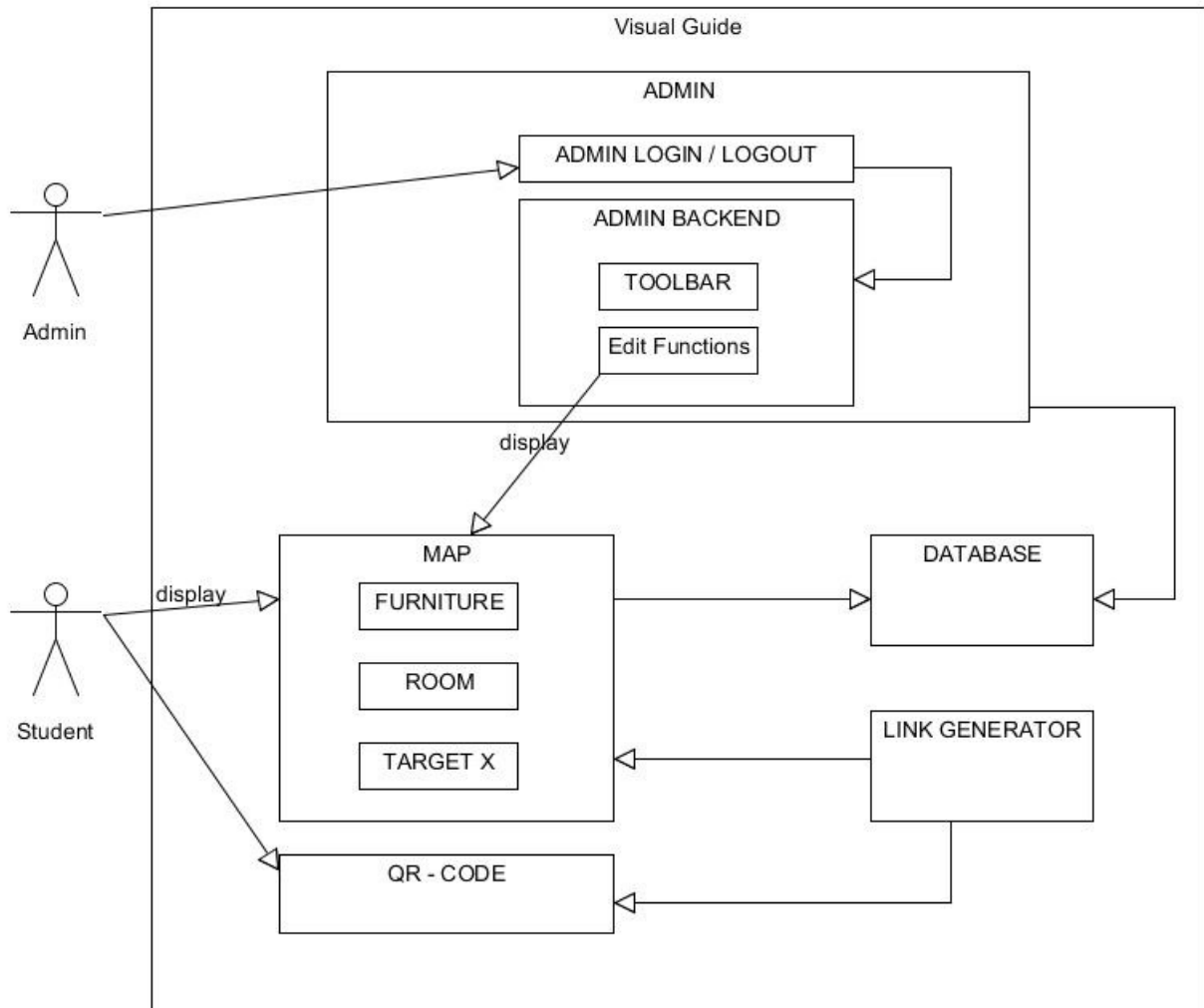
The font size 12 makes it easy to read for everyone.

Corporate Design Colors	
	#E78F08 - Background of headlines
	#1C94C4 - Font color of buttons
	#FFFFFF - Background color
	#EEEEEE - Background of buttons
	#CCCCCC - Background of content
	#000000 - Font color
	#4C96A8 - Furniture color
	#FF6600 - Shelf color

The table on the right hand side displays the hexadecimal color code for most of the used colors.

### 3.2 System architecture model

The following architecture model provides a view on the system as a compilation of modules.



The administrator is able to login and to logout of the admin backend. After login he is also able to access the toolbar and the edit functions.

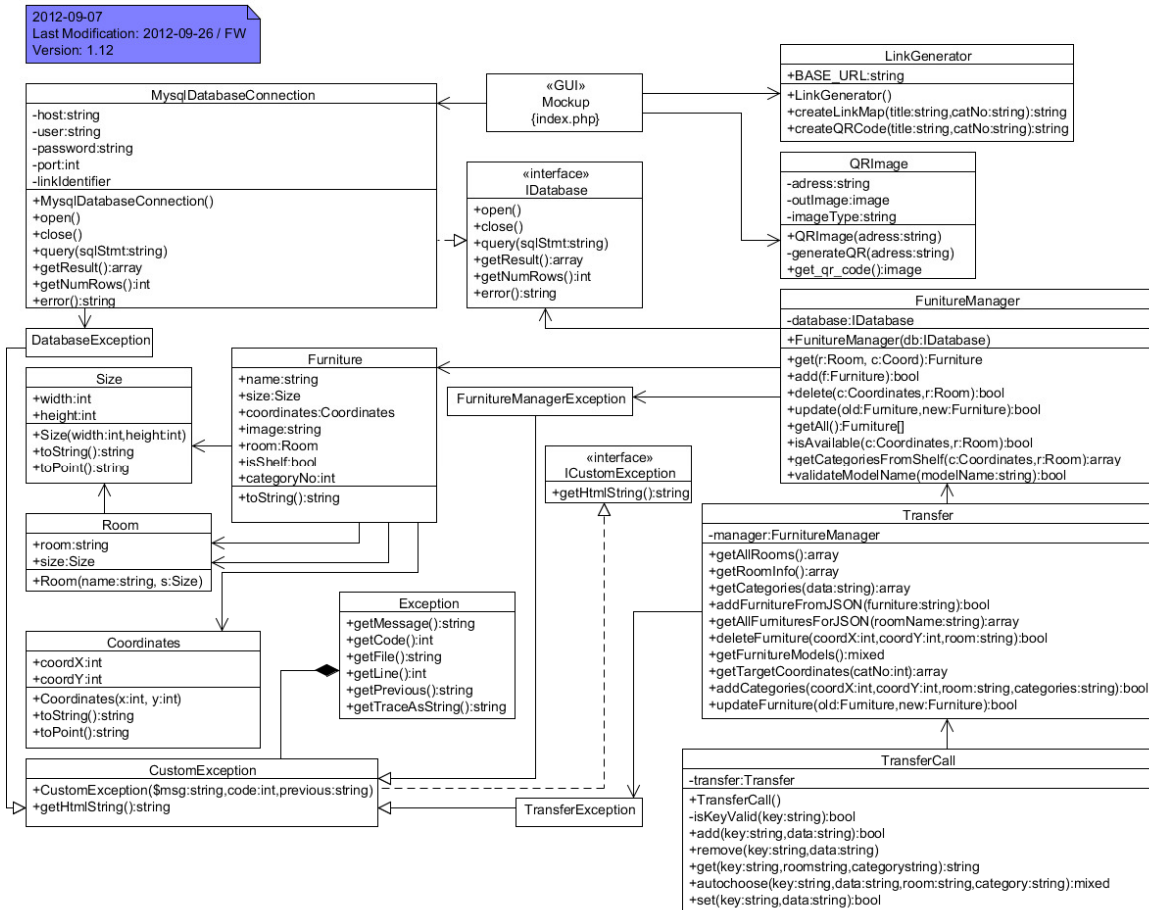
The administrator backend relies at several points on the database connection which is required for the backend functions to work correctly.

The student is able to display a QR-code or to open the Visual Guide map.

The Visual Guide map consists of the sections “Furniture”, “Room” and “Map”. Moreover the map also has a direct access to the database which is required to display the components.

## 4. Implementation

### 4.1 Class diagram



### 4.2 The purpose of the classes

#### MySqlConnection

Reads access data for a MySQL database connection from a configuration file. It wraps essential PHP functions to control a database connection and to do queries. It implements an interface “IDatabase” to be easily changed for different database systems.

#### DatabaseException, CustomException, TransferException, FurnitureManagerException

To ease error handling, class specific exceptions are used which derive from the class CustomException which derives from the PHP class Exception. Thus the existence of the class Exception is a precondition for the derived class CustomException to work. Those custom exceptions are able to output error messages as an HTML-formatted string. CustomException is implemented against an interface “ICustomException” to ensure uniformity.

#### Room



A simple container for the room's attributes as there are room name and a Size object.

#### Size

A simple container for a width and a height attribute.

#### Furniture

This class represents all values a piece of furniture: The name of the furniture's model, a reference to an instance of the Size, Coordinates and Room class, the filename of the furniture image and all category numbers assigned to the shelf.

#### LinkGenerator

The LinkGenerator class has methods to generate the links dependent on the supplied parameter. It generates either a link to the Visual Guide or encodes the link as a QR-code image relying on the QRImage class.

#### QRImage

This class is an extension to an Open Source class that takes a string and generates a QR-code containing the encoded string.

#### FurnitureManager

The FurnitureManager class contains all the methods which directly communicate with the database class. It retrieves furniture data from the database and builds furniture objects from it and vice versa. There are methods to update or to delete furniture objects. It is possible to get all categories by supplying a shelf identifier (coordinates and room name). There are also methods to determine if an location is already occupied by another furniture and to validate a model name.

#### Transfer

The Transfer class contains functions which directly puts data into a format that is suitable for transferring to JavaScript. There are methods which delegate the furniture actions to the FurnitureManager and there are methods which directly access the database to retrieve data about rooms, for instance. One certain method gets a category number and calculates the position of the target "X" with data retrieved from the database.

#### TransferCall

This class validates the post data that is being transferred from the client side. There are also methods to validate the call itself and to categorize the call. This class is invoked by a simple script "transfermanager.php" that is being called by a jQuery AJAX request.





### 4.3 The program's structure

The core of the application was modeled as object-oriented as possible for reasons of clarity and comprehensibility, to provide reusability and to make the program easy to maintain. The data that is transferred internal between parts of the program is given a structure this way. Furthermore an object-oriented programming style makes the program more robust and secure.

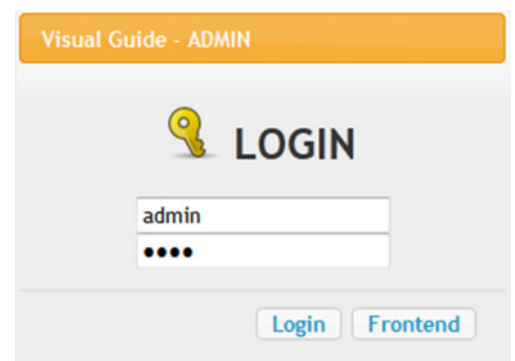
For instance, inside the core of the program the classes “Size”, “Furniture” and “Room” are frequently used by different neighbor classes.

The class “MysqlDatabaseConnection” was implemented against a so-called “interface” which is like a pattern for a class and its methods. This enables to use a different database than SQL without doing many changes to the source code.

### 4.4 Functions

#### 4.4.1 Secure administrator login

Security in the administrator login is provided by using sessions. With these a verification of the login is possible before each single access to the “TransferCalls” class and so prevent unauthorized access or changes by functions which are reserved for administration purposes. A session is valid for 2 hours. 15 Minutes before expiration a warning is shown in the admin backend.



#### 4.4.2 Mock-Up

As the “ding2/Artesis” system is not fully functional without the book search services provided by DBC it cannot be analyzed to develop the system. It is also not possible to demonstrate the functionality of the Visual Guide.

However, DBC provided a data extraction of the Horsens University library.

Thus a simple mock-up script was developed that provides basic functionality to demonstrate the usage of the Visual Guide.

The data was imported into a separate database that is not a part of the Visual Guide system.

The mock-up page is opened by the user in a browser and offers a search box to look up books from the mock-up data. To ease looking up the mock-up data an autocomplete-function offers suggestions after typing three letters.

In the search results both a link to the Visual Guide and a link to display a QR-code are displayed.

#### VIA Library Mockup Page

Book title:	Leg
-------------	-----

---

**Book result**

---

Title: Leg der styrker motorikken (0-3 år)  
Author: Winter, Vibeke  
Type: Bog  
Location: 79.3  
University: horsens

MAP: [Show MAP](#)  
QR-Code: [Show QR-Code](#)

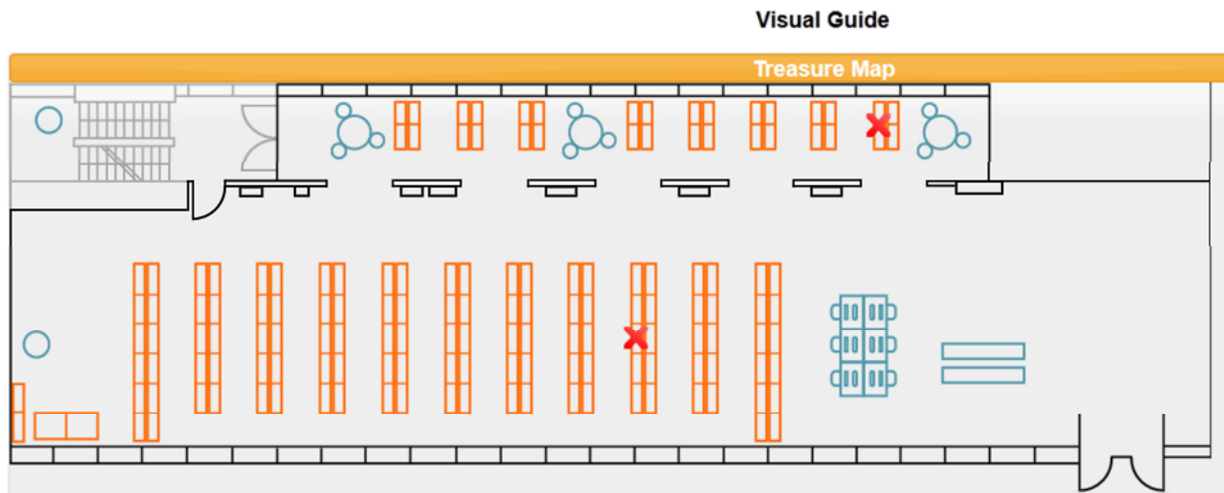
---

Title: Legeplads  
Author:



#### 4.4.3 Visual Guide

The link which refers to the “visualmanager.php” in the “/mockup” directory contains the book title, the category number and a “display” value as parameter. The “display” value indicates whether to display the Visual Guide or the QR-code.



#### 4.4.4 Process of constructing the map

The ground plan of the room is set as background image of the main container element.

At first the positions and the graphics of furniture is fetched from the database by PHP. The Furniture will be positioned relatively to the position of the ground plan.

In PHP code the assignment of categories to shelves is determined from the database. The coordinates of the target “X” are calculated by the shelf position and the shelf measures and are set to the middle of the shelf.

One or more target “X” coordinates and the associated room are returned.

By jQuery these coordinates are being processed and the ground plan of the according room is loaded as background image. One or more target “X” are put on the map relatively to the position of the ground plan.

#### 4.4.5 Process of generating a QR-code

For generating QR-codes an Open Source library was used. The input string complies to the link to the Visual Guide with parameters. The QR-code is returned as an image. The QR-code is displayed on the screen to be scanned by a smartphone using a QR-app. The smartphone will recognize the string as an internet address and will open the web browser to display the Visual Guide.

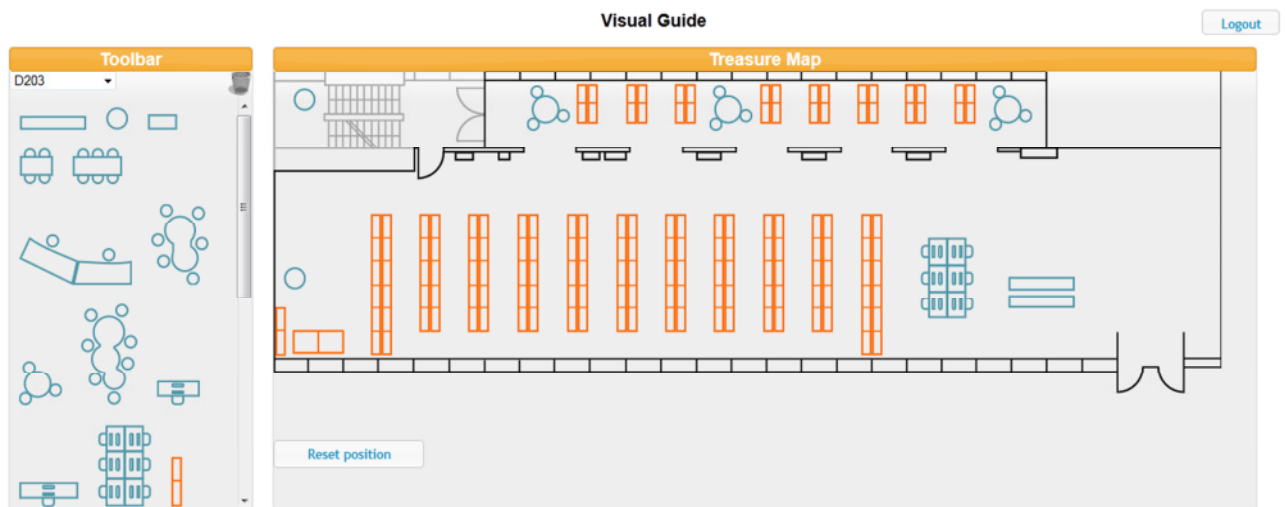


## 4.4.6 Admin-Backend

### 4.4.6.1 Login Process

Login is possible at “/admin/index.php” using the given login data. The default login data can be obtained from the attachment on page 2.

A session is started which guarantees safety from unauthorized access to the admin functions. By clicking the logout button the admin is logged out and the session is destroyed.



### 4.4.6.2 Process of choosing a room

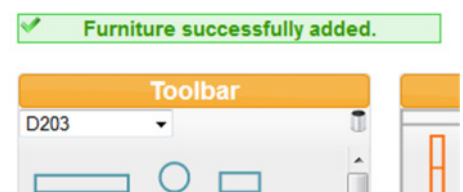
In the backend there is a toolbar on the left including a dropdown field to choose a room to edit.

As the project only covers room D203 the other rooms are saved in the database but cannot be administrated. Therefore the rooms aren't selectable in the dropdown field.

If those rooms have a consistent storage system someday, it only will be necessary to change a few lines in the source code.

### 4.4.6.3 Process of adding new furniture

There is a listing in the toolbar of all available furniture models. They can be placed new in the map by left-clicking. The furniture is saved automatically after insertion.



### 4.4.6.4 Process of deleting furniture

By clicking the recycle bin, a red delete button appears on each piece of furniture on the map. Clicking the button removes the furniture permanently.



#### 4.4.6.5 Process of moving furniture

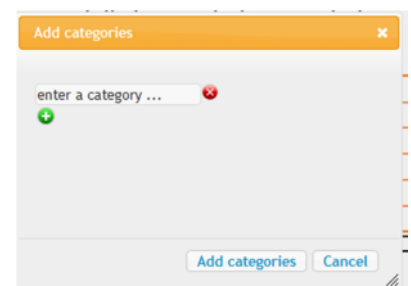
A piece of furniture can be moved within the map by a drag&drop operation with a left-click. There is a 10x10 pixel sized grid that defines the minimal difference of the new position to the old position.

#### 4.4.6.6 Process of resetting a position

Resets each furniture's position to the position that it had when it appeared on the map.

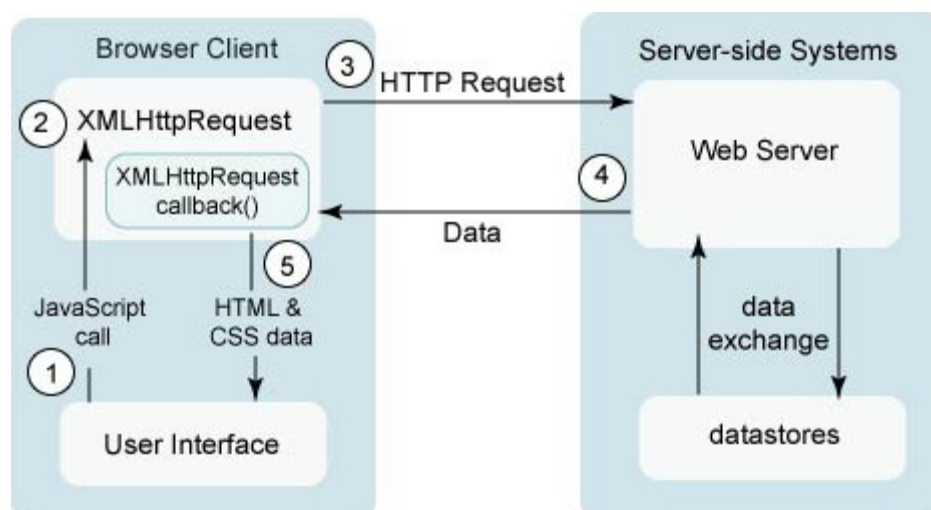
#### 4.4.6.7 Process of changing categories

A single left-click on a piece of furniture opens a window in which the corresponding category numbers are displayed and can be changed, deleted or added.



#### 4.4.7 Operating principles of jQuery / AJAX

The interface for the data exchange between jQuery and PHP encodes the transmitted data as JSON-strings which are being transferred using AJAX. On the Server/PHP side the data processing is being initiated by a script “transfermanager.php” which invokes the class “TransferCalls”.



The call is being initiated by a user-controlled event, for instance adding a piece of furniture to the map. The JSON-encoded data is transferred from the client to the server by the http-request.

The server decodes and processes the data and returns it JSON-encoded back to the client. On the client it is applied to the map using jQuery.



An example for an AJAX-request in jQuery syntax:

```
$.ajax({
    url: 'transfermanager.php',
    type: 'post',
    dataType: 'json',
    data: 'key=addCategories&data=' + JSON.stringify(dataObject),
    success: function(response) {
        alert(response);
    },
    error: function() {
        $.error("ajax error");
    }
});
```

The call “addCategories” including the parameter “dataObject” is being encoded to JSON implicitly and transmitted to the PHP script “transfermanager.php”.

The relevant section in “transfermanager.php”:

```
if (isset($_POST["key"])) {
    $key = $_POST["key"];
    $room = (isset($_POST["room"])) ? $_POST["room"] : "";
    $category = (isset($_POST["category"])) ? $_POST["category"] : "";
    $data = (isset($_POST["data"])) ? $_POST["data"] : "";
    $call = new TransferCall();
    $response = $call->autochoose($key, $data, $room, $category);
    echo json_encode($response);
} else {
    echo "42 KEY PARAM NOT SET";
}
```

On server side the data input is done in PHP by reading the \$\_POST array for entries. The transmission of the JSON-encoded return value is done by a simply “echo” output.

On client side AJAX will receive the output and call it “response”. The JSON-decoding will happen implicit. On not empty return values “success” is asserted, on empty return values “error” is asserted. In the example above a dialog containing the value of “response” is opened in case of success, otherwise an exception saying “ajax error” is thrown to the JavaScript console.

A list of the available transfer calls can be found in appendix on page 2.

## 4.5 URL scheme

When generating links always three parameters “title”, “catNo” and “display” are used.

“title” contains the URL-encoded book title. For encoding in PHP the predefined command urlencode() respectively for decoding urldecode() can be used.

“catNo” contains a single category number to mark on the map with an target “X” which should not need to be URL-encoded.



“display” tells the firstly invoked PHP script whether to generate and output an QR-code encoded link to the mobile version of Visual Guide or to redirect directly to the desired desktop version of Visual Guide.

The following example is taken from the “visualManager.php” in the “/mockup” directory. In the mockup system the links in the search results have a third parameter “display” which can contain the value “qr” or “map”.

The first links displayed in the search results looked like this:

<http://server.dk/vg/mockup/visualManager.php?title=A+book+title&catNo=321.4&display=qr>  
<http://server.dk/vg/mockup/visualManager.php?title=A+book+title&catNo=321.4&display=map>

To display a QR-code in the browser a function must be written in PHP that includes the class “QRImage.class.php” and invokes it this way with the mobile URL as parameter.

So the first link referred to the shown PHP document “visualManager.php” where the parameter “display” is being evaluated.

Dependent on this parameter, two different PHP files can be addressed next.

If a link to the desktop version of the Visual Guide is desired, a redirect to the “index.php” in the source code’s root directory happens.

If a link to the mobile version of the Visual Guide is desired, the “mobile.php” in the source code’s root directory has to be encoded as an QR-code.

In both cases the parameter “title” and “catNo” are added to the new URL.

Examples for the generated links:

Link for redirection to the desktop version:

<http://www.anyserver.com/vg/index.php?title=A+book+title&catNo=321.4>

Link encoded as QR-code:

<http://www.anyserver.com/vg/mobile.php?title=A+book+title&catNo=321.4>

```
require_once 'mockup.global.inc.php';
require_once PATH_CLASSES . 'QRImage.class.php';

if (isset($_GET["display"]) && isset($_GET["title"])
    && isset($_GET["catNo"]))
{
    $choice = $_GET["display"];
    $title = $_GET["title"];
    $catNo = $_GET["catNo"];
    if (!empty($title) && !empty($catNo)) {
        if ($choice == 'qr') {
            $title = urlencode($title);
            $url = 'http://';
            $url .= $_SERVER['SERVER_NAME'];
            $url .= '/mobile.php';
            $url .= "?title=" . $title;
            $url .= "&catNo=" . $catNo;
            $qr = new QRImage($url);
            echo $qr->get_qr_code();
        } else if ($choice == 'map') {
            $url = '../index.php';
            $url .= "?title=" . $title;
            $url .= "&catNo=" . $catNo;
            header('Location: ' . $url);
        } else {
            echo "INVALID COMMAND";
        }
    }
} else {
    echo "INVALID COMMAND";
}
```



## 5 Testing

In advance the decision was made to use Unit Tests as software testing method. Unfortunately all attempts to install free PHP Unit Test tools failed due to software errors which we were unable to resolve. A short extract of the error messages can be found in appendix on page 5.

Testing must not be omitted as it is a significant part of the software development process. Testing classes and functions is mandatory to eliminate security flaws. Validation has always been an important part in the project implementation.

As we were unable to get the software to run in a certain amount of time, it was decided to be best to skip this part and to write new testing classes specifically for this project. These custom testing classes evaluate some of the most important core functions.



Methods to test	Input value	Assertion	Response	Success
TransferCall->isKeyValid(\$key)	„addFurniture“	True	True	Yes
TransferCall->setPosition(\$key,\$data)	„setPosition“,0	False	False	Yes
TransferCall->addCategories(\$key,\$data)	„addCategories“, { "furniture": { "coordX":251,"coordY":147,"room": "D203"}, "categories":["61.01","33.00"]}‘	True	True	Yes
TransferCall->removeFurniture(\$key,\$data)	'deleteFurniture',{ "coordX":150,"coordY":147,"room": "D203"}	True	<b>False</b>	<b>No</b>
Room->getRoom()	shelf1	shelf1	shelf1	Yes
Room->getSize()	12	12	12	Yes
Size->getHeight()	5	5	5	Yes
Size->getWidth()	42	42	42	Yes
LinkGenerator->createLinkMap(\$title,\$catNo)	"Psykofarmaka", 61.642	visualManager.php?title=Psykofarmaka&catNo=61.642&display=map	visualManager.php?title=Psykofarmaka&catNo=61.642&display=map	Yes
LinkGenerator->createQRCode(\$title,\$catNo)	Null,Null	""	""	Yes
FurnitureManager::add(\$furniture)	"abc",Size(42, 42)	False	False	Yes
FurnitureManager:: get(\$coord, \$room)	Coordinates(23, 3), Room("abc");	Null	Null	Yes
FurnitureManager::isAvailable(\$coord,\$room)	Coordinates(80, 45), Room("def")	False	False	Yes
FurnitureManager::deleteFurniture(\$coord, \$room)	Coordinates(80,30), Room("D203"))	True	<b>False</b>	<b>No</b>

Conclusion of the test:

Almost all tests concluded satisfyingly and the testing scenario could be finished successfully with one exception.

The deleteFurniture respectively the removeFurniture method failed due to of an unknown reason. There was no time left to fix this bug.



## 5.1 Documentation of the PHP classes

Documentation was generated automatically using the classes and method's comments by the software "ApiGen". The documentation files are stored in the path "PHP\_DOC\testUnit".

## 6 Conclusions

After the library software which was ought to be modified showed to be unaccessible, we decided to develop for a new system called "ding2 / Artesis". An separate database is needed as there was no official interface.

For the software design the latest web developing techniques like PHP, jQuery and HTML5 were used. The whole web design has been highly customized.

Data processing is mostly done in PHP, the dynamic frontend part is done in jQuery.

The program was programmed object-oriented for reasons of clarity and comprehensibility, to provide reusability and to make the program easy to maintain.

A mock-up system was entirely created just for the purpose of demonstrating the Visual Guide.

The map itself contains several pieces of furniture which are retrieved from the database. The target X is placed by knowing where category numbers are positioned. You can either call the map directly or generate an QR-code that redirects to an mobile version of the Visual Guide for smartphones.

In the administrator backend the librarians are able to login and to add, delete move or changing categories of a room's furniture.

The URL scheme of a link to the Visual Guide is like this:

*<http://server.dk/vg/mockup/visualManager.php?title=A+book+title&catNo=321.4&display=map>*

As Unit Tests, the testing method of choice, did not work due to third-party-software errors, specialized test classes were created for the project.

Before the Visual Guide can be used productively, an ding2/Artesis system must be installed on a server that offers full access to the developer. Categories have to be entered in the admin interface manually once. Best for the university would be to have a contact person for technical purposes who is able to maintain the library software.





## 7 Sources, references and literature

### Books:

Kai Laborenz: *CSS-Praxis*. Galileo Computing, 2008

Carsten Möhrke: *Besser PHP programmieren*. Galileo Computing, 2012

Gunnar Thies, Stefan Reimers: *PHP 5.3 + MySQL 5.1: Der Einstieg in die Programmierung dynamischer Websites*. Galileo Computing, 2009

Andreas Vdovkin: *jQuery – kurz und gut*. O'Reiley, 2009

### Internetpages-/sources:

<https://biblioteket.ucsj.dk/site>: The UCSJ using Ding2/Artesis. September 27th, 2012

[http://www.dbc.dk/produkter-services/artesis\\_produktoversigt](http://www.dbc.dk/produkter-services/artesis_produktoversigt): Introduction about Artesis. September 27th, 2012

<http://ting.dk>: The TING concept group developing the Ding.TING system. September 27th, 2012

<http://ting.dk/node/119>: Required web services to run the Ding.TING system. September 27th, 2012

<http://ting.dk/blog/ding2-kodebasen-overgives-til-ting-community>: Annunciation of Ding2 as Open-Source continuation of Ding.TING. September 27th, 2012

<https://github.com/ding2/ding2>: Source Code of Ding2 with a hint that it's known as "Artesis". September 27th, 2012

<http://viauc.stg.ding.dbc.dk>: Ding2 / Artesis test system by DBC. September 27th, 2012

<http://www.swetake.com>: QR-code generator class. September 27th, 2012



## 8 Appendices

Page 2: Admin login data

Page 2: Semantical irreducible sentences

Page 2: List of transfer calls

Page 3: Database model

Page 4: Ding.TING System Architecture

Page 5: PHPUnit failure

Page 6: System requirements

Page 6: Integrated development environments

Page 7: Installation of ding2/Artesis