

## **Betriebliche Projektarbeit**

### *Neuprogrammierung des eyekit-Statistikmoduls*

Bericht zur betrieblichen Projektarbeit von  
**Christian Herberger**  
IHK-Prüfungsnummer: 30681

als **Fachinformatiker**  
Fachrichtung  
**Anwendungsentwicklung**

Prüfungsbetrieb: **eyeworkers interactive GmbH**  
Durmshaimerstr. 55  
76185 Karlsruhe

Betreuer: **Tim Riedel**

Durchführungszeitraum: 2.4.2009 - 17.4.2009

## Betriebliche Projektarbeit

### ***Ausbildungsberuf***

---

Fachinformatiker, Fachrichtung Anwendungsentwicklung

### ***Projektbezeichnung***

---

Neuprogrammierung des Statistikmoduls des eyeKit

### ***Auszubildender***

---

Christian Herberger  
Wilhelm-Hauff-Weg 8  
75015 Bretten

Telefon: (07252) 42843  
E-Mail: [christian.herberger@iai.fzk.de](mailto:christian.herberger@iai.fzk.de)

IHK-Prüfungsnummer: 30681

### ***Ausbildungsbetrieb***

---

Forschungszentrum Karlsruhe GmbH  
Institut für Angewandte Informatik  
Hermann-von-Helmholtz-Platz 1  
76344 Eggenstein-Leopoldshafen

### ***Ausbilder***

---

Dipl.-Ing. Jürgen Engelmann  
Telefon: (07247) 82 5752  
E-Mail: [juergen.engelmann@iai.fzk.de](mailto:juergen.engelmann@iai.fzk.de)

## 1. Inhaltsverzeichnis

1. Inhaltsverzeichnis.....	II
2. Ausgangssituation.....	- 1 -
2.1. Projektumfeld .....	- 1 -
2.2. Projektbeschreibung .....	- 1 -
2.3. Projektziel .....	- 1 -
2.4. Prozessschnittstellen .....	- 1 -
2.5. Technisches Umfeld .....	- 2 -
3. Projektplanung .....	- 2 -
3.1. Ist-Zustand .....	- 2 -
3.2. Soll-Zustand.....	- 2 -
3.3. Zeitliche Planung .....	- 3 -
3.4. Ablaufplan.....	- 4 -
3.5. Kostenplanung.....	- 4 -
4. Projektdurchführung.....	- 5 -
4.1. Entwicklungsumgebung.....	- 5 -
4.2. Änderungen am bestehenden System .....	- 5 -
4.3. Gestaltung der Benutzeroberfläche .....	- 5 -
4.4. Erstellen des Dateibrowsers.....	- 6 -
4.5. Darstellung der Zugriffsliste .....	- 7 -
4.6. Speichern/Laden-Funktion.....	- 9 -
4.7. CSV-Export .....	- 10 -
5. Testphase .....	- 11 -
5.1. Test der Benutzeroberfläche.....	- 11 -
5.2. Test des Dateibrowsers .....	- 12 -
5.3. Test der Listendarstellung .....	- 12 -
5.4. Test der Speichern/Laden-Funktion .....	- 13 -
5.5. Test des CSV-Exports .....	- 14 -
6. Projektabschluss .....	- 14 -
6.1. Ist/Soll-Vergleich.....	- 14 -
6.2. Änderungen der Zeitplanung.....	- 14 -
6.3. Mögliche Verbesserungen.....	- 14 -
7. Quellenangaben .....	- 15 -
8. Anhang A: Benutzerhandbuch .....	I
8.1. Allgemeines.....	I
8.2. Darstellen der Liste.....	I
8.3. Speichern.....	I
8.4. Laden .....	II
8.5. CSV-Export .....	II
9. Anhang B: Glossar.....	III

## 2. Ausgangssituation

### 2.1. Projektumfeld

Die Werbeagentur eyeworkers interactive hat 18 Beschäftigte. Sie bietet Webdesign, das eigene CMS eyeKit als Backend für eine Webpräsenz, Design von Printmedien und Ideen/Gestaltung von Geschäftsausstattungen u.ä. an.

### 2.2. Projektbeschreibung

Das eyeKit enthält ein Statistikmodul (eyeStats), über welches die User sich anzeigen lassen können, wie oft auf welche Unterseite zugegriffen wurde. Derzeit ist diese Anzeige nur für einzelne Unterseiten möglich und die Zahlen erstrecken sich vom ersten Zugriff auf die Webpräsenz bis heute. Außerdem werden zur Darstellung IFrames, eine veraltete Technologie, verwendet. Im Rahmen dieses Projektes soll die Technik erneuert und die Funktionalität erweitert werden.

### 2.3. Projektziel

Das Ziel dieses Projektes ist es, das eyeStats-Modul um genauere Auswahlkriterien, aktuellere Technik und neue Speichern- und Ladenfunktionen zu erweitern.

### 2.4. Prozessschnittstellen

- Tim Riedel [tim.riedel@eyeworkers.de](mailto:tim.riedel@eyeworkers.de) Auftraggeber, Ansprechpartner
- Torsten Koch [torsten.koch@eyeworkers.de](mailto:torsten.koch@eyeworkers.de) Ansprechpartner

## 2.5. *Technisches Umfeld*

Die Werbeagentur eyeworkers interactive nutzt einen Subversion-Server, um neue Versionen ihres CMS einander zugänglich zu machen. Zum Programmieren wird eine lokal installierte Version des eyeKit aufgesetzt, welche auf einem XAMPP-Lite läuft. Es werden PHP 5.2.8, MySQL 5.1.30, Apache 2.2.11 (jeweils für Windows) und ext JS 2.2 eingesetzt. Es handelt sich hierbei nicht um die neusten Versionen, wodurch sie repräsentativer für die Server der Kunden sind.

Zum Programmieren wird eclipse Ganymede mit den Plugins phpEclipse und svnEclipse genutzt.

## 3. Projektplanung

### 3.1. *Ist-Zustand*

Das derzeitige eyeKit bietet ein Statistikmodul an, welches Seitenzugriffe und Klicks anzeigen kann, doch derzeit ist die Funktionalität extrem eingeschränkt. So stört Kunden, dass man nur alle Daten seit Erstellung der Webpräsenz anzeigen lassen kann, und dass es nur möglich ist einzelne Unterseiten, jedoch nicht ganze Bereiche, auszuwählen. So ist das Modul nur bedingt hilfreich, da der Betreiber nicht nachvollziehen kann, wann die meisten Zugriffe stattgefunden haben und er auch nicht ohne größeren Aufwand feststellen kann, welcher Seitenbereich der beliebteste ist.

### 3.2. *Soll-Zustand*

Im Rahmen des Projekts wird die Anzeige der Klicks und Visits neu programmiert. Folgende Anforderungen werden gestellt:

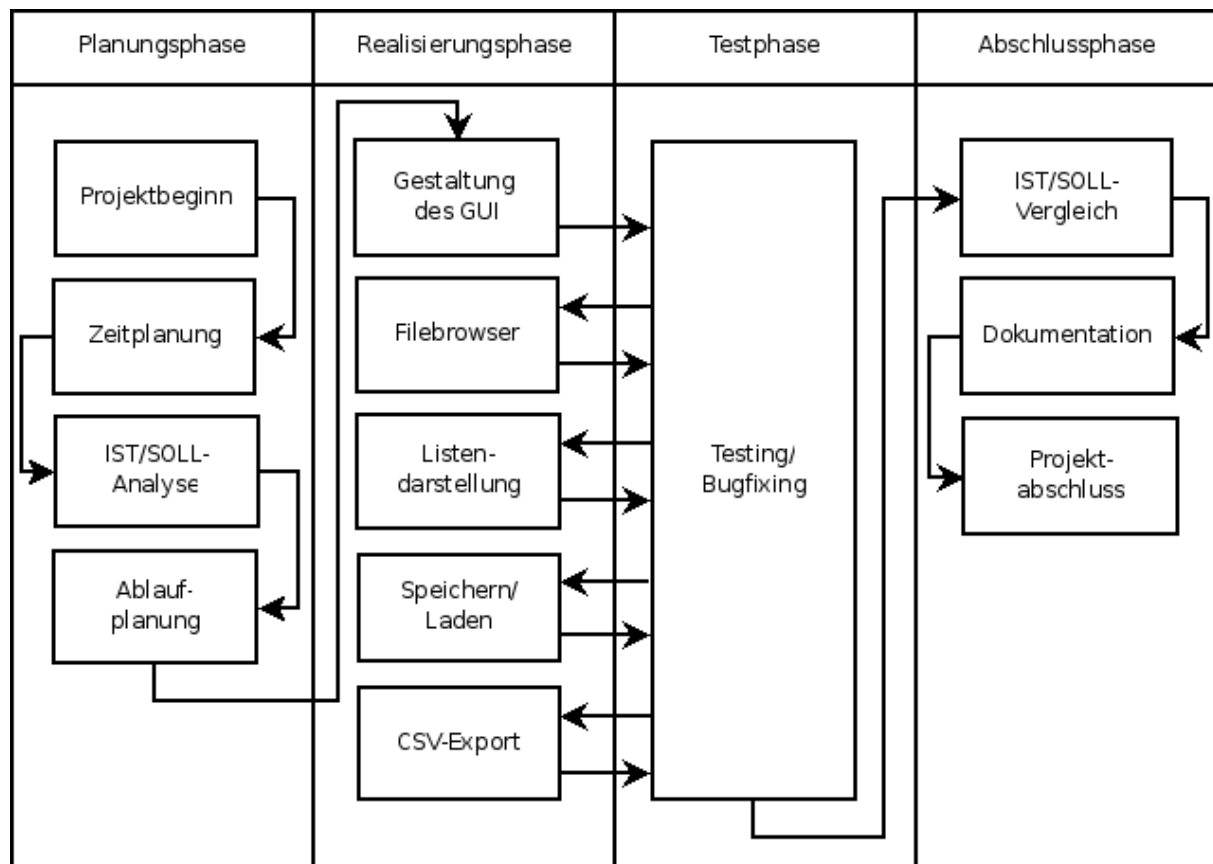
- Datumsauswahl für die Visits, sodass der Kunde sich auf bestimmte Zeiträume beschränken kann
- Die Dateiauswahl soll nun auch mit Ordnern möglich sein
- Der Benutzer soll seine Datei-/Ordner- und Datums-/Zeitraumwahl speichern und laden können
- Die erzeugte Liste soll als CSV-Datei exportierbar sein
- Anstelle veralteter Technologie soll extJS zum Einsatz kommen
- Die Datenauswertung soll performant sein, außerdem soll für den Benutzer erkennbar sein, wann der Webserver bereit ist
- Die gewählten Daten sollen über den gesamten Zugriffszeitraum auf den Administrationsbereich verfügbar sein, d.h. nicht gelöscht werden wenn der Benutzer eine andere Unterseite aufruft

### 3.3. Zeitliche Planung

Tätigkeit	Zeitaufwand	Datum
<b>Planungsphase</b>		
Sichtung der Applikation	1 Stunde	2. April
Konzeption auf Basis des Anforderungenkatalogs	9 Stunden	2.-3. April
<b>Realisierungsphase</b>		
Umsetzung	20 Stunden	3.- 7. April
<b>Testphase</b>		
Testing/Bugfixing	25 Stunden	8.-15. April
<b>Abschlussphase</b>		
Dokumentation	15 Stunden	15.-17. April

Damit wird die vorgesehene Zeit von 70h erreicht.

### 3.4. Ablaufplan



Die Testphase läuft parallel zur Umsetzungsphase, da für die einzelnen zu programmierenden Funktionen die fehlerfreie Anwendung der vorgehenden Funktionen notwendig ist.

### 3.5. Kostenplanung

Für die verwendete Software fallen keinerlei Kosten an, da mit eclipse, apache, PHP und MySQL OpenSource-Software verwendet wird und für das extJS in der Agentur eine Lizenz vorliegt.

Es wird keine spezielle Hardware benötigt, da das Projekt durch die gegebene Software lokal auf einem beliebigen PC lauffähig ist.

Es fallen keine Personalkosten an, da über die Dauer des Projekts die normalen Vertragsbedingungen des FZK gelten.

## 4. Projektdurchführung

### 4.1. Entwicklungsumgebung

Das Projekt wird auf einem PC mit Windows XP als Betriebssystem entwickelt, da dieser direkt verfügbar war. Auf diesen wurden eclipse und XAMPP-Lite heruntergeladen und installiert.

Für XAMPP-Lite musste der ServerRoot geändert werden, um auf den Projektpfad von eclipse zuzugreifen.

Für eclipse wurden anschließend phpEclipse und svnEclipse über die automatische Updatefunktion von eclipse installiert. Über svnEclipse wurde schließlich vom Fileserver von eyeworkers die neuste Version des eyeKit über den SVN-Befehl update heruntergeladen.

### 4.2. Änderungen am bestehenden System

Für jede im Rahmen des Projekts erstellte Datei muss im eyeKit in der Hauptdatei des Adminbereiches ein switch-case-Konstrukt erweitert werden, damit der Benutzer die neuen Dateien über das eyeKit aufrufen kann und beim Programmieren auf bestehende Session-Variablen zugegriffen werden kann. Außerdem musste die Configdatei des eyeStats-Modul erweitert werden, dies wird später näher betrachtet.

### 4.3. Gestaltung der Benutzeroberfläche

Die Benutzeroberfläche (GUI) sollte soweit wie möglich mit dem bestehenden eyeKit übereinstimmen, damit der übliche Workflow für den Benutzer erhalten bleibt. Zu diesem Zweck werden bestehende CSS-Klassen verwendet, welche für die verwendeten div-Container Rahmenfarben, Grundausrichtung und eine Zuordnung des Designs zur spätern Funktion ergeben.

Damit die sonstigen eyeKit-Funktionen im neuen eyeStats-Modul aufgerufen werden können, muss der Kopfbereich aus einer der bestehenden eyeStats-Dateien herauskopiert werden. Damit werden z.B. ein automatisches Log und die Navigation eingefügt.



Als nächstes werden die Elemente gebraucht, die zur Auswahl und Anzeige der Daten benötigt werden. Dazu werden die vorher erstellten div-Container mit extJS-Panels gefüllt. Diese ermöglichen auf einfache Weise, Inhalt dynamisch einzubinden.

Gebraucht werden ein TreePanel für den Dateibrowser, ein FormPanel für die Datumswahl und ein normaler div-Container ohne ext-Panel für die Anzeige der Daten. Dem TreePanel und dem Anzeige-Div werden über Javascript dynamisch von der Größe des Browserfensters abhängig Höhen- und Breitenangaben zugewiesen und bei Größenanpassung des Fensters angeglichen. Dadurch wird kein Platz verschwendet und alle wichtigen Daten sind dauerhaft sichtbar.

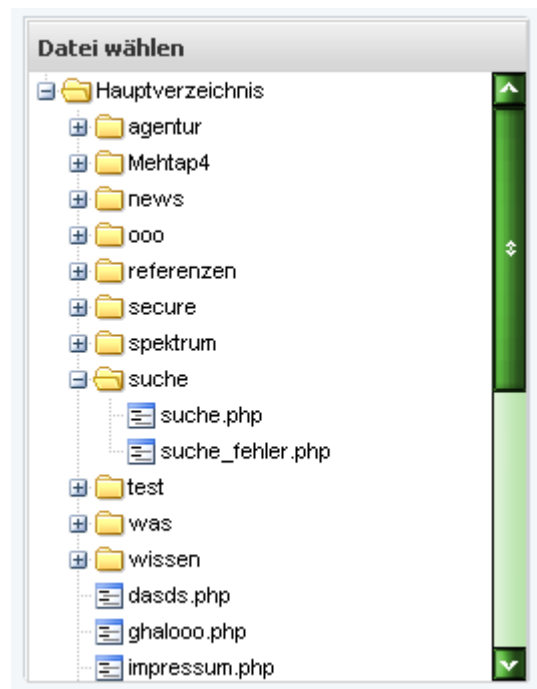


#### 4.4. Erstellen des Dateibrowsers

Als nächstes wird der Dateibrowser gebraucht, über den die Dateien bzw. Ordner, für welche die Statistik erstellt werden soll, ausgewählt werden können.

Als erstes muss entschieden werden, wie die Ordnerstruktur erstellt werden soll. Als Alternativen stehen zur Verfügung:

- Auslesen der tatsächlichen Ordnerstruktur
- Auslesen der Statistik-Datenbank und berechnen der entsprechenden Hierarchie
- Erstellen einer neuen Tabelle in der Datenbank, in welcher jeder Ordner und jede Datei mit Eltern-Element als einzelner Datensatz existiert, und bauen des Filetrees aus dieser Tabelle



Da im Rahmen des Projektes auf die Performance geachtet wird, wäre die 3. Möglichkeit die effektivste, jedoch darf hier nichts an der bestehenden Tabellenstruktur geändert werden, um die Abwärtskompatibilität mit älteren eyeKit-Versionen, die Kunden im Einsatz haben, zu erhalten.

Möglichkeit 2 wäre nicht sehr performant, da die Pfade in der Datenbank nicht in einer einfach auszulesenden Variante gespeichert sind und außerdem die Datenbank nicht mit dem Filesystem übereinstimmen muss.

Nach Ausschluss der anderen beiden Möglichkeiten wird die Variante gewählt, das tatsächliche Filesystem auszulesen. Da in diesem jedoch der Ordner, in welchem die Administrationsdateien des eyeKit liegen, und ein Ordner zum Bearbeiten von noch nicht veröffentlichten Dateien vorhanden sind, müssen diese (und weitere, von Kunden festgelegte Ausnahmen) aus dem Dateibrowser ausgeblendet werden.

Zu diesem Zweck wird die bestehende Konfigurationsdatei des eyeStats-Moduls um ein Array erweitert, in welchem Kunden Ordner (abhängig vom Webserver-Root) festlegen können, welche dann ausgeblendet werden.

Dem TreePanel des ext-Frameworks kann man auf einfache Weise eine Quelldatei angeben, welche intern über Ajax aufgerufen wird. Gibt man in dieser Datei einen JSON-String aus und übermittelt diesen an das Treepanel, wird automatisch der entsprechende Zweig generiert. Da die Zweige dynamisch geladen werden, reicht eine Funktion, welche den Inhalt eines einzigen Ordners ohne Beachten der Unterordner als JSON zurückliefert. Dadurch kann selbst ein relativ großer Filetree performant erzeugt werden, da geringe Datenmengen ausgelesen und übermittelt werden müssen.

#### 4.5. *Darstellung der Zugriffsliste*

Mit den Datumswählern und dem Dateibrowser kann man nun die Parameter für die Datenbankabfrage auswählen. Mit dieser werden nun die Daten aus der Datenbank abgerufen. Hierbei muss beim Datei-Parameter nach 3 verschiedenen Möglichkeiten getestet werden und jede dieser Möglichkeiten extra behandelt werden:

- Index-File (index.php/html) gewählt: Aus der Datenbank müssen nicht nur die Daten für die Datei abgerufen werden, sondern auch für den darüberliegenden Ordner, da bei Aufruf dessen die Indexdatei vom Webserver an den Browser ausgeliefert wird. Die beiden Datensätze müssen addiert werden, um die tatsächlichen Zugriffszahlen auf die Datei zu erhalten.
- Beliebige andere Datei gewählt: Aus der Datenbank werden einfach die Zugriffszahlen auf die Datei geholt und übermittelt.
- Ordner gewählt: Hier müssen alle Dateien innerhalb des Ordners ermittelt werden und für diese die Zugriffszahlen ermittelt werden. Dabei muss man darauf achten, ob sich eine Index-Datei im Ordner befindet, und wie oben die Zugriffe auf den übergeordneten Ordner dazugezählt werden. Bei der Ordnerwahl wird außerdem die Summe aller ermittelten Zahlen berechnet und zusätzlich zu den anderen Daten übermittelt, da dies dem Wunsch einiger Kunden entspricht.

Bei allen Abfragen muss man beachten, ob es in der Datenbank tatsächlich Daten gibt, und wenn nicht, muss ein Datensatz mit 0 Clicks und 0 Visits angezeigt werden.

Jeder Datensatz für die Anzeige setzt sich aus folgenden Komponenten zusammen: Datei - Clicks - Visits. Die Clicks sind hierbei alle Aufrufe einer Datei, die Visits sind die Besucher, bestimmt über die IP, gültig für 30 Minuten. Das heißt, wenn ein Besucher innerhalb von 30 Minuten mehrmals die gleiche Unterseite besucht, werden die Clicks immer erhöht, die Visits jedoch insgesamt nur einmal.

Hierbei sind die Clicks in der Datenbank einfach als Zahl gespeichert, welche der Datei zugeordnet ist. Jedes mal, wenn die Datei aufgerufen wird, wird diese Zahl um 1 erhöht.

Da diese Zahl keinerlei Zuordnung zu den Zugriffsdaten hat, kann diese nicht in Abhängigkeit der gewählten Daten abgerufen werden, sondern ist generell die Gesamtzugriffszahl seit Veröffentlichung der Seite.

Die Visits werden auf eine andere Art berechnet. Für jede IP gibt es in der Datenbank eine Liste der Reihenfolge der Dateizugriffe. Aus dieser Liste müssen die Dateinamen herausgefiltert werden, um die Visits auf jede Datei zählen zu können. (Hierbei muss bei Index-Dateien beachtet werden, dass Datensätze, in welchen sowohl die Index-Datei als auch der übergeordnete Ordner vorhanden sind, nicht doppelt gezählt werden)

Da in den Datensätzen der Zugriffsreihenfolgen die Start-Daten aufgeführt sind, kann man diese in Abhängigkeit der gewählten Daten abrufen. Für den Benutzer mag irritierend sein, dass die Clicks ohne Datumsabhängigkeit und die Visits mit dargestellt werden, doch für weitere Versionen des eyeKit sind große Änderungen an der Datenbank geplant, wodurch dieses Problem gelöst werden kann - im Verlauf des Projekts ist (wie schon weiter oben geschrieben) keine Änderung der bestehenden Tabellenstruktur der Datenbank erlaubt, weswegen im Rahmen des Projekts nicht darauf geachtet werden kann.

Der Abruf der Daten aus der Datenbank erfolgt wie beim Dateibrowser asynchron, was wiederum einen JSON-String zurückliefert. Aus diesem String wird über Javascript eine HTML-Tabelle berechnet und in den Anzeige-Div eingefügt. Damit ist bei der ganzen Funktionalität kein kompletter Reload der Seite nötig, was die Geschwindigkeit der Zugriffe erhöht.

Suchergebnisse für /referenzen: (als CSV herunterladen)		
URL	Clicks (gesamt)	Visits (zwischen 28.01.2005 und 28.04.2009)
/referenzen/audi.php	281	254
/referenzen/basi.php	229	216
/referenzen/berres.php	202	191
/referenzen/weisenburger.php	198	185
<b>Summe</b>	<b>910</b>	<b>846</b>

#### 4.6. Speichern/Laden-Funktion

Als nächstes sollen die gewählte Datei und der Zeitbereich, welcher über die DatePicker berechnet wird, gespeichert und geladen werden.

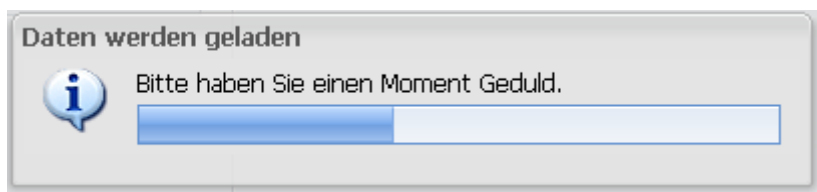
Zuerst wird zum Speichern ein zusätzlicher Button zu dem Formular, mit welchem die beiden DatePicker liegen, hinzugefügt. Wird dieser aufgerufen, wird asynchron, also ohne Seitenreload, eine Funktion aufgerufen, welche die Datei, die Eigenschaft, ob diese Datei eine Datei oder ein Ordner ist, und der gewählte Zeitraum in Tagen in Abhängigkeit des Benutzers gespeichert.

Hierbei wird in der eigentlichen Speicherfunktion eine neue Tabelle in der Datenbank angelegt, jedoch nur, wenn diese noch nicht existiert. Diese Tabelle stellt kein Problem dar, da sie einfach aufgebaut ist und nicht mit bestehenden Daten kollidiert. Damit sind die Daten persistent, also dauerhaft, gespeichert und die Abwärtskompatibilität bleibt gegeben.

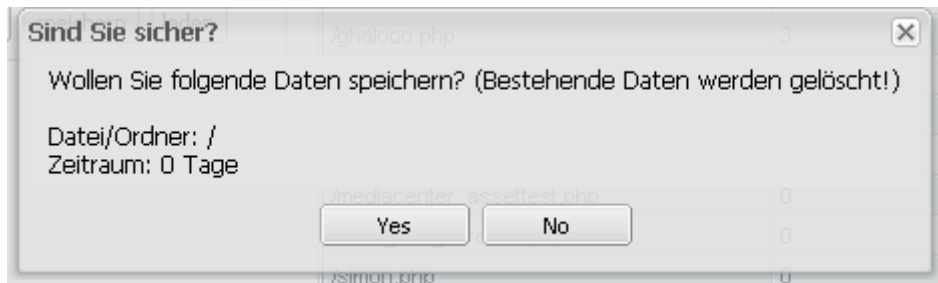
Zum Laden wurde überlegt, wann die gespeicherten Daten abgerufen werden sollen. Auf Nachfrage wurde überlegt, dass man zwar direkt beim Aufrufen des Statistikmoduls die Datenbank nach gespeicherten Daten prüfen könnte, doch das wäre nicht unbedingt das Verhalten, welches ein Benutzer erwartet.

Daher wird der Benutzeroberfläche ein weiterer Button für einen weiteren asynchronen Funktionsaufruf hinzugefügt. Mit diesem wird die Datenbank nach den gespeicherten Daten (wiederum in Abhängigkeit des Benutzers) abgefragt, und mit diesen Daten wird anschließend die Liste der Zugriffe geholt.

Damit der Benutzer nicht von der Wartezeit, in der anscheinend nichts passiert, nicht erstaunt ist, wird ein Ladebalken hinzugefügt, welcher zeigt, wann die Abfrage fertig ist.



Außerdem wird vor dem Speichern und Laden jeweils eine Sicherheitsanfrage aufgerufen, damit der Benutzer nicht aus Versehen durch einen falschen Klick seine Daten überschreibt. In der Sicherheitsabfrage beim Speichern werden außerdem die eingegebenen Daten (Zeitraum, gewählte Datei) noch einmal angezeigt, damit der Benutzer diese noch einmal überprüfen kann.



#### 4.7. CSV-Export

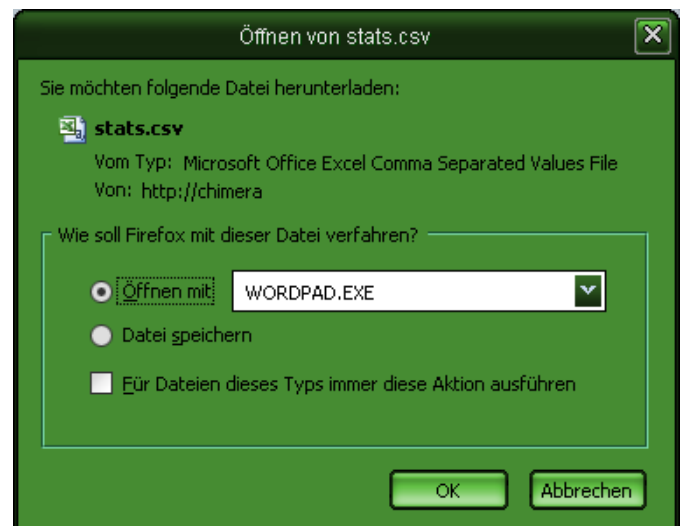
Eine weitere Funktion, die von Kunden gewünscht wurde, ist, dass man die Statistik aus dem eyeKit exportieren können soll.

Damit die Kompatibilität mit möglichst vielen Programmen, die diese Tabellen nutzen können, gegeben ist, wird ein standardisiertes Dateiformat verwendet: CSV (Comma-Separated Values)

In diesem Format wird für jeden neuen Datensatz eine neue Zeile genutzt und die einzelnen Werte des Datensatzes sind (wie der Name des Formates sagt) durch Kommata voneinander getrennt. CSV ist somit nützlich für viele kurze Daten, doch nicht für Volltext-Einträge, da in diesen jederzeit ein Komma auftauchen kann, was als Wert-Trennzeichen interpretiert wird.

Da in Dateinamen und Pfadangaben Kommata theoretisch verboten sind, eignet sich das CSV-Format sehr gut dafür.

Zum Herunterladen dieser Datei wird zur Datenanzeige ein Link hinzugefügt, welcher ein entsprechendes Script ausführt. Dieses Script ist nicht asynchron, vermeidet jedoch trotzdem den PAGERELoad, da im Header der zurückgegebenen Datei festgelegt wird, dass sie als Download angeboten wird. Außerdem legt der Dateiheader ebenfalls den Dateinamen fest, da dieser sonst eyeKit.php, nach der aufgerufenen Datei, heißen würde.



Das Script entspricht dem Script zur Listendarstellung, nur dass anstelle der HTML-Tabelle die CSV-Darstellung berechnet wird.

Beispiel-CSV-Datei:

```
/referenzen/audi.php,281,0  
/referenzen/basi.php,229,0  
/referenzen/berres.php,202,0  
/referenzen/weisenburger.php,198,0  
Summe,910,0
```

## 5. Testphase

Im Rahmen des Projektes gibt es keine eindeutige Testphase, in welcher das ganze Script getestet wird, dafür wird nach jeder neu programmierten Funktion diese getestet.

### 5.1. *Test der Benutzeroberfläche*

Um die korrekte Darstellung der Benutzeroberfläche und die automatische Größenanpassung zu testen, wird die Seite in folgenden Browsern getestet:

- Internet Explorer 6/7/8
- Firefox 3
- Safari
- Opera

Hierbei wird als einziges Betriebssystem Windows XP verwendet, da bis auf verschiedene Schriftarten das Browserverhalten in verschiedenen Betriebssystemen gleich bleibt. Da die verwendete Version des extJS-Framework keinen Support für Chrome liefert, bleibt dieser Browser ungetestet.

Auf Tests ohne Javascript und Flash wurde verzichtet, da das eyeKit schon bei der Benutzeranmeldung prüft, ob diese Technologien vorhanden sind, und dem Benutzer ggf. die Anmeldung verweigert.

Alle Tests waren für die getesteten Browser fehlerlos.

## 5.2. *Test des Dateibrowsers*

Um die Funktionalität des Dateibrowsers zu testen, werden die dargestellten Daten mit dem tatsächlichen Dateisystem verglichen, außerdem wird geprüft ob der erzeugte Suchparameter die korrekte Form erhält und ob die Sperrung von speziellen Ordnern und Dateien über die Konfiguration funktionieren.

Bei diesem Test wird erkennbar, dass die verwendeten TreePanel-Funktionen des extJS-Frameworks bei der Wahl eines Ordners oder einer Datei außer dem Root-Verzeichnis des Webservers einen zusätzlichen Slash am Anfang erzeugt.

Erklärbar ist dieser Slash dadurch, dass der Root als Slash dargestellt wird und als Trennzeichen zwischen den Ordner Ebenen ebenfalls ein Slash genutzt wird. Umgangen wird dieses Verhalten durch Ersetzen von // durch /, bevor der String zum Abrufen der Daten verwendet wird.

Einfach den Slash zu entfernen hätte nicht gereicht, da man in diesem Fall das Root-Verzeichnis nicht als Suchpfad hätte wählen können, denn dies hätte einen leeren String als Parameter erzeugt.

Nach dem Fix für den doppelten Slash gab es innerhalb des Dateibrowsers keine weiteren Fehler.

## 5.3. *Test der Listendarstellung*

Der Test der Listendarstellung erfolgt während des eigentlichen Programmierens, da hier die eigentliche Funktion in mehrere Unterfunktionen unterteilt ist, welche aufeinander aufbauen.

Zuerst wird der AJAX-Request getestet, indem ein kleiner Request auf eine PHP-Datei erfolgt, welcher die Such-Parameter zurückgibt. Werden diese nicht angezeigt, befindet sich ein Fehler im Code. Dieser Test funktionierte fehlerfrei.

Als nächstes werden die Datenbankaufrufe, die die Clicks abrufen, getestet. Da diese als Zahl in der Datenbank gespeichert sind, ist dies relativ unproblematisch, außer dem vorher erwähnten Verhalten bei Index-Dateien. Dieses kann im Falle der Clicks durch eine einfache Erweiterung der MySQL-Abfrage um die Abfrage der Daten für den übergeordneten Ordner erreicht werden.

Nachdem die Clicks korrekt dargestellt werden, müssen nun die Visits getestet werden. Diese Datenbankabfrage ist komplizierter und brachte einige Probleme, doch nachdem die ganze Funktion gelöscht und mit anderer Unterscheidung der einzelnen Fälle (siehe Auflistung in Kapitel *Darstellung der Zugriffsliste*, Seite 6) neu entwickelt wurde, wurden diese Probleme auf einen simplen Denkfehler reduziert.

Dieser Fehler bestand darin, dass bei Index-Files die Zahl der Datensätze mit Zugriff auf den übergeordneten Ordner zu den Zugriffen auf das Index-File addiert wurde, wodurch das Ergebnis zu groß war. Die Lösung besteht darin, dass bei der Datenbankabfrage schon sowohl nach der Index-Datei als auch dem Ordner geprüft werden muss, wodurch Clickwege, in welchen beide Einträge vorhanden sind nicht mehr doppelt gezählt werden.

Außerdem wurde hier erkennbar, dass für Dateien, die zwar im Filesystem existieren, die aber noch keine Zugriffe hatten, ein Datensatz mit 0 Zugriffen zurückgeliefert werden muss. Diese Dateien sollen dargestellt werden, da sie schon im Dateibrowser vorhanden sind und der Benutzer verwirrt werden würde, wenn sie in der Liste fehlen.

Damit funktioniert die Datenbankabfrage fehlerfrei und damit werden die richtigen Daten an die eigentliche Darstellungsfunktion geliefert.

Für die Darstellung der Daten wird getestet, ob die CSS-Klassen richtig angewendet und die übermittelten Daten richtig ausgewertet werden.

Hierbei wird erkennbar, dass extJS beim Decodieren von JSON-Strings mit Arrays automatisch eine Methode hinzufügt, welche das Löschen von einzelnen Arrayelementen vereinfachen soll. Im Rahmen des Projekts wird also bei der Darstellung der Daten darauf geachtet, dass diese Methode nicht als *undefined* in der Tabelle auftaucht.

#### 5.4. *Test der Speichern/Laden-Funktion*

Es wird getestet, ob die Daten im richtigen Format gespeichert werden, was beim Laden passiert, und was passiert, wenn die Tabelle noch nicht bzw. schon vorhanden ist und was mit bereits vorhandenen Daten passiert.

Hier werden alle vom Projektleiter gewünschten Funktionen erfüllt:

- Die Tabelle wird erstellt, wenn sie noch nicht vorhanden ist, ansonsten passiert nichts
- Vorhandene Daten werden überschrieben
- Beim Laden wird direkt die Liste mit diesen Daten dargestellt

Außerdem ist das Format der gespeicherten Daten korrekt, damit ist diese Funktion fehlerfrei.

### 5.5. *Test des CSV-Exports*

Da hier die Funktion der Listendarstellung wiederverwendet wird, wird nur die Darstellung der Daten überprüft. Diese funktioniert fehlerfrei.

Aufgrund der Tatsache, dass in einem Dateipfad keine Kommata vorkommen dürfen, kann es durch diese an falschen Stellen nicht zu Fehlern kommen.

## 6. Projektabschluss

### 6.1. *Ist/Soll-Vergleich*

Es wurde ein neues Statistik-Modul für eyeKit entwickelt, wobei alle verlangten Funktionen erhalten sind. Damit ist das Projektziel erfüllt. Die Neuentwicklung des Dateibrowsers kann in anderen Modulen des eyeKit wiederverwendet werden und die Nutzung der AJAX-Technologie anstelle des IFrames kann ebenfalls dank modularem Aufbau weiterhin verwendet werden.

### 6.2. *Änderungen der Zeitplanung*

Da die Durchführungs- und die Testphase parallel abliefen, dabei aber zusammen so lange dauerten wie geplant, kam es zu keinen Änderungen.

### 6.3. *Mögliche Verbesserungen*

In einem der nächsten Releases des eyeKit könnten die Tabellen zur Statistikerfassung neu gestaltet werden, wodurch das eyeKit-Modul performanter und genauer arbeiten könnte.

## 7. Quellenangaben

- PHP-Dokumentation (Englisch)  
<http://php.net/manual/en>
- MySQL-Dokumentation (Englisch)  
<http://dev.mysql.com/doc/>
- SelfHTML - Javascript-Dokumentation (Deutsch)  
<http://de.selfhtml.org/javascript/index.htm>
- extJS API Dokumentation (Englisch)  
<http://extjs.com/depoy/dev/docs/>

## 8. Anhang A: Benutzerhandbuch

### 8.1. Allgemeines

Für alle hier nicht aufgeführten Funktionen gilt, dass diese im eyeKit-Benutzerhandbuch erklärt werden und für das Projekt irrelevant sind.

### 8.2. Darstellen der Liste

Um eine Statistik abzurufen, führen Sie folgende Schritte aus:

1. Datei im Dateibrowser auswählen - mit Klick auf das + neben einem Ordner gelangen Sie eine Ebene tiefer.
2. Datum auswählen: Dies erfolgt über die beiden "DatePicker". Klicken sie auf das Symbol neben dem Eingabefeld, um den Kalender zu öffnen, und wählen Sie das gewünschte Datum aus.
3. Bestätigen: Klicken Sie nun auf den "Statistik zeigen"-Button, damit die Daten aus der Datenbank geholt und als Tabelle dargestellt werden. Die Tabelle erscheint auf der rechten Seite des Fensters.

### 8.3. Speichern

Um Ihre gewünschten Suchparameter abzuspeichern, sind die gleichen Schritte wie zur Anzeige der Liste nötig, nur dass Sie anstelle "Statistik zeigen" auf den Button "speichern" klicken. Anschließend werden Ihre gewählten Parameter angezeigt und Sie werden noch einmal gefragt, ob Sie diese Daten wirklich speichern wollen. Wählen Sie "nein" zum Abbrechen oder "ja", damit die Parameter in die Datenbank eingetragen werden.

Es werden nicht die genauen Start- und Enddaten gespeichert, sondern der Zeitraum zwischen den beiden gewählten Daten.

**Achtung:** Bereits vorhandene Daten werden überschrieben, daher bestätigen Sie nur, wenn sie sich wirklich sicher sind.

## 8.4. Laden

Um Ihre gespeicherten Daten abzurufen, klicken Sie einfach auf den Button "laden". Nun erscheint eine Abfrage, ob Sie wirklich laden möchten. Bestätigen Sie mit "ja", wird sofort die Liste abgerufen für die gespeicherte Datei mit den Daten, welche ab heute so viele Tage zurückliegen, wie Sie vorher gespeichert haben.

Beispiel:

### Auswahl beim Speichern:

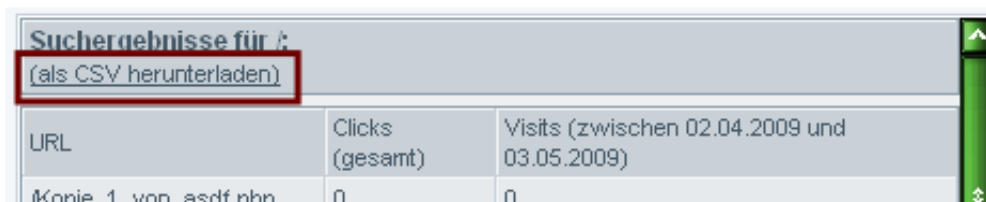
Startdatum: 5.3.2009  
Enddatum: 5.4.2009  
Zeitraum: 31 Tage

### Ergebnis beim Laden am 3.5.2009

Zeitraum: 31 Tage  
Enddatum: 3.5.2009  
Startdatum: 2.4.2009

## 8.5. CSV-Export

Um eine Liste als CSV-Datei herunterzuladen, müssen Sie zuerst die entsprechende Tabelle anzeigen lassen und anschließend auf den Link "(als CSV herunterladen)" oberhalb der Tabelle klicken. Damit wird Ihnen die Datei "stats.csv", welche der dargestellten Tabelle entspricht, zum Download angeboten. Sie können diese direkt mit einem Programm zur Weiterverarbeitung öffnen oder sie auf der Festplatte bzw. Wechseldatenträgern speichern.



Suchergebnisse für :		
<a href="#">(als CSV herunterladen)</a>		
URL	Clicks (gesamt)	Visits (zwischen 02.04.2009 und 03.05.2009)
.Kopie 1 von asdf.php	0	0

## 9. Anhang B: Glossar

Begriff	Erklärung
<i>eyekit</i>	Das Firmeneigene CMS von eyeworkers
<i>CMS</i>	Content Management System: Ein System, welches es ermöglicht, dass mehrere Benutzer ohne HTML/Programmierenkenntnisse eine Homepage erstellen können
<i>SVN</i>	Subversion: Ein Dienst zur Versionsverwaltung. Kann mehrere Versionen gleichzeitig verwalten und später zu einem Hauptrelease zusammenführen (=mergen)
<i>XAMPP-Lite</i>	Eine kleinere Version von XAMPP, einem Softwarepaket bestehend aus Apache, MySQL, PHP und Perl
<i>Apache</i>	Webserver-Programm
<i>MySQL</i>	Weit verbreitetes Datenbanksystem
<i>PHP</i> <i>Perl</i>	Serverseitige Scriptsprachen (d.h. sie werden auf dem Server ausgeführt und das Ergebnis wird an den Client weitergegeben), welche oft im Web eingesetzt werden (Im Rahmen des Projektes wird PHP verwendet)
<i>extJS</i>	Framework für AJAX, basierend auf JavaScript
<i>AJAX</i>	Asynchronous JavaScript and XML, ermöglicht Zugriffe auf serverseitige Skripte, ohne die ganze Seite neu laden zu müssen
<i>JavaScript</i>	Clientseitige Scriptsprache (d.h. der Code wird an den Client gesendet und dieser führt diesen aus), wird oft im Web für dynamische Effekte auf Webseiten eingesetzt
<i>eclipse</i>	Programmeditor, ursprünglich für Java entwickelt, kann er durch Plugins erweitert werden (im Rahmen des Projekts werden phpEclipse und svnEclipse verwendet)
<i>OpenSource</i>	Quelloffen, d.h. jeder kann den Quellcode des Programmes herunterladen und verändern
<i>CSS</i>	Cascading Stylesheet: Auszeichnungssprache, um Webseiten besser zu gestalten und Layout und Gestaltung voneinander zu trennen
<i>DatePicker</i> <i>TreePanel</i> <i>FormPanel</i>	extJS-Klassen, welche einfach füllbare Container erzeugen
<i>Root-Verzeichnis/ Webserver-Root</i>	Verzeichnis, in welchem alle Dateien und Ordner des Webauftritts liegen.
<i>JSON</i>	JavaScript Object Notation: Ein speziell formatierter String, welcher einfach aus einem Objekt einer beliebigen Programmiersprache erzeugt und genauso einfach wieder zurückgewandelt werden kann. Wird verwendet, um Daten, welche über AJAX erzeugt wurden, zu übermitteln

<i>CSV</i>	Comma Seperated Values: Format, um Datensätze auf einfach weiterzuverarbeitende Art zu Speichern. Hierbei ist jede Zeile ein Datensatz und die Werte werden durch Kommata voneinander getrennt
<i>(Datei-)Header</i>	Metadaten (= Daten, die der Benutzer nicht sieht, aber für den PC eine Bedeutung haben) am Anfang einer Datei



Name, Vorname: **Herberger, Christian**  
(in Druckbuchstaben)

Diese Erklärung ist bei Einreichen der Dokumentation als letztes Blatt anzuhängen! Bitte bei jeder der 4 Dokumentationen beifügen, wobei mindestens 1 Ausfertigung mit Originalunterschriften versehen sein muss (restliche Ex. als Kopie möglich).

**Persönliche Erklärung  
zur Projektarbeit und Dokumentation  
im Rahmen der Abschlussprüfung in IT-Berufen**


Ich versichere durch meine Unterschrift, dass ich die betriebliche Projektarbeit und die dazugehörige Dokumentation selbstständig in der vorgegebenen Zeit erarbeitet habe.

Ich habe keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet.

Eggenstein, 4.5.2009  
Ort, Datum

C. Herberger  
Unterschrift des Prüfungsteilnehmers

Zur Kenntnis genommen:

  
Forschungszentrum Karlsruhe GmbH  
In der Helmholtz-Gemeinschaft  
IPS / Abteilung Berufliche Ausbildung  
Postfach 3840, 76021 Karlsruhe  
Tel.: 0 72 47 82 51 80 / 25 46  
Fax: 0 72 47 82 22 30  
Ausbilder/-in + Firmenstempel