

# Software Requirements Specification

Lehrveranstaltung IT Projektarbeit

## Polaroid Fotoclub Webseite

Ausgeführt von: David Wolf, Leonhardt Schwarz, Tom Schalbar, Christoph Müllner

Begutachter: Veronika Winter

Wien, 12. Juni 2014

---

Version	Datum	Change
0.1.0	2014/02/28	Erste Erstellung des Dokuments
0.1.1	2014/03/19	ERD & Mockups wurden hinzugefügt

Tabelle 1: Dokumentenentwicklung

## **Inhaltsverzeichnis**

## 1 Vision und Kurzbeschreibung des Projekts

Eine responsive Webseite zum Anzeigen und Archivieren von Fotos soll mit modernen Webtechnologien realisiert werden.

**Ziel:** Implementation einer Webseite welche sowohl auf Desktop PCs als auch auf Mobiltelefonen in kontinuierlichem Design angezeigt wird. Ein Nutzer soll einen Account anlegen können und Fotos auf die Webseite hochladen können. Andere Nutzer sollen dann in der Lage sein diese Fotos zu bewerten und kommentieren zu können. Um die Privatsphäre des Nutzers zu wahren können mithilfe einer Freundeverwaltung Fotos nur bestimmten Nutzern zugänglich gemacht werden.

## 2 Überblick über die geforderte Funktionalität

- Anlegen und verwalten von Accounts
- Hochladen von Fotos
- Responsive/ Fluid Design
- Albumverwaltung
- Fotos bewerten, kommentieren und gegebenenfalls dem Administrator melden
- Administratorbereich
- Statistiken zu hochgeladenen Fotos (Viewcount)
- Freundeverwaltung
- Einstellungen zur Privatsphäre (Nur Freunde können bestimmte Fotos sehen)
- Suchfilter

## 3 Systemabgrenzung und detaillierte Funktionale Anforderungen

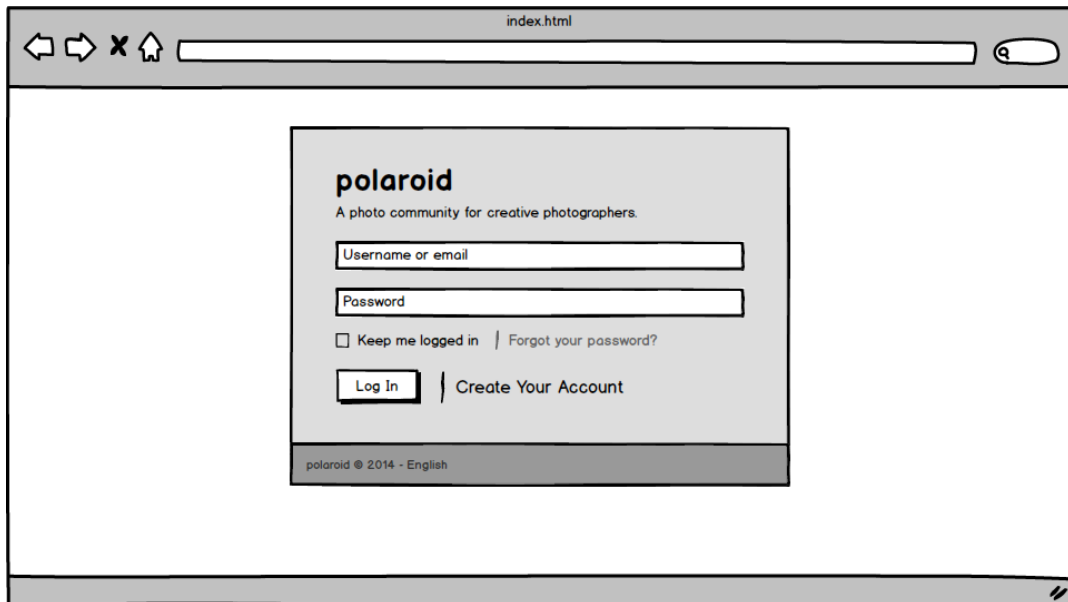
### 3.1 Node.js

Node.js ist eine serverseitige Plattform zum Betrieb von Netzerkanwendungen. Insbesondere lassen sich Webserver damit realisieren. Node.js basiert auf der JavaScript-Laufzeitumgebung „V8“, die ursprünglich für den Chrome-Browser entwickelt wurde und bietet daher eine ressourcensparende Architektur, die eine besonders große Anzahl gleichzeitig bestehender Netzwerkverbindungen ermöglicht

## 4 Schnittstellen

### 4.1 Benutzerschnittstellen

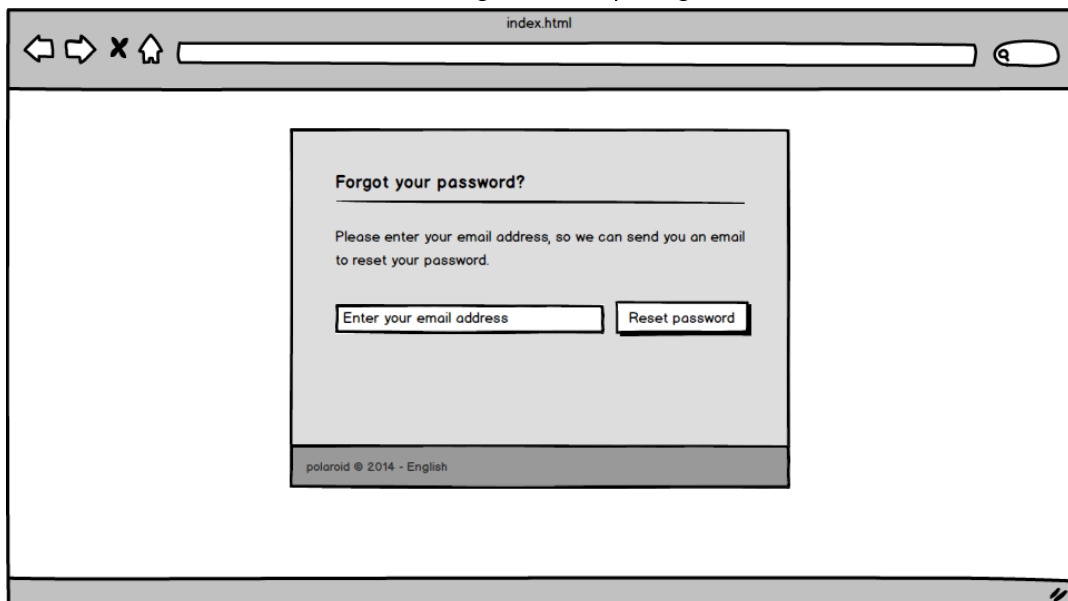
#### 4.1.1 Webseite



A browser window mockup showing a login page for 'polaroid'. The browser's address bar contains 'index.html'. The page features a central login form with the following elements:

- polaroid** logo
- Slogan: A photo community for creative photographers.
- Input field: Username or email
- Input field: Password
- Form elements: ☐ Keep me logged in | [Forgot your password?](#)
- Buttons: **Log In** and **Create Your Account**
- Footer: polaroid © 2014 - English

Abbildung 1: Mockup: Login



A browser window mockup showing a 'Forgot your password?' page for 'polaroid'. The browser's address bar contains 'index.html'. The page features a central form with the following elements:

- Forgot your password?** title
- Text: Please enter your email address, so we can send you an email to reset your password.
- Input field: Enter your email address
- Button: **Reset password**
- Footer: polaroid © 2014 - English

Abbildung 2: Mockup: Forgot your password

The mockup shows a web browser window with the address bar displaying 'index.html'. The main content area contains a 'Sign up' form. The form has a title 'Sign up' followed by a horizontal line. Below the line are six input fields: 'First Name', 'Last Name', 'Username', 'Your Email', 'New Password', and 'Re-enter Password'. A 'Sign up' button is positioned below the 'Re-enter Password' field. At the bottom of the form container, there is a footer text: 'polaroid © 2014 - English'.

Abbildung 3: Mockup: Registrierungsformular

The mockup shows a web browser window with the address bar displaying 'User Startpage'. The page has a navigation bar with a 'Log' button and links for 'Home', 'Gallery', 'Me', and 'Find Friends'. On the right side of the navigation bar are icons for search, settings, and a user profile. Below the navigation bar is a section titled 'News Feed'. The feed contains two identical entries. Each entry starts with a placeholder icon (a square with an 'X') and the text 'Max Mustermann - 24 minutes ago'. To the right of each entry are icons for 'Like' (thumbs up) and 'Comment' (speech bubble), followed by the counts 'Like (64)' and 'Comment (10)'. The content area of each entry is a large rectangle with a diagonal 'X' across it, indicating a placeholder for an image or video.

Abbildung 4: Mockup: News Feed - Seite

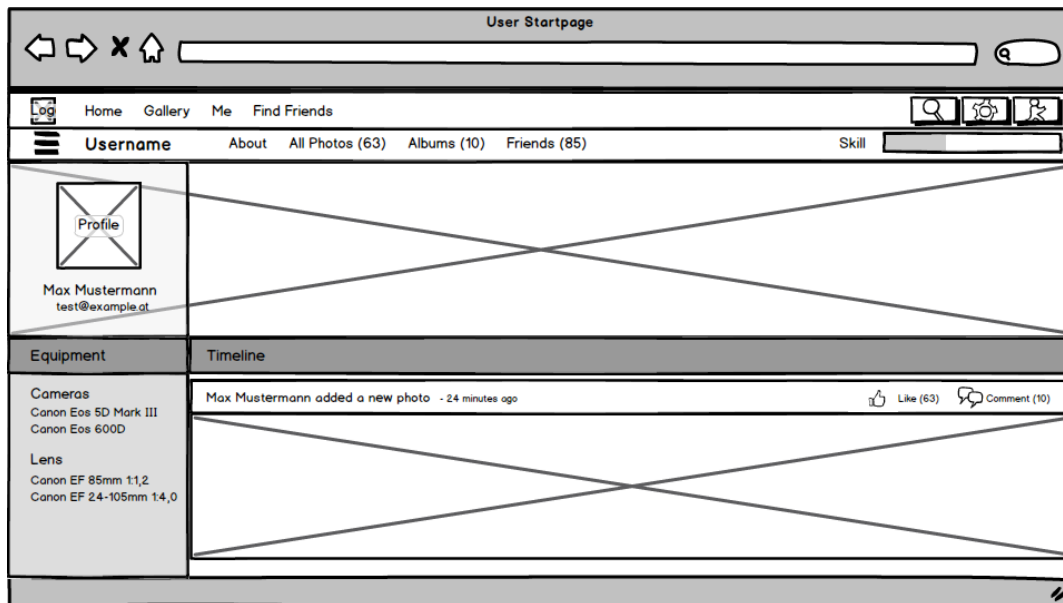


Abbildung 5: Mockup: Profil - Seite

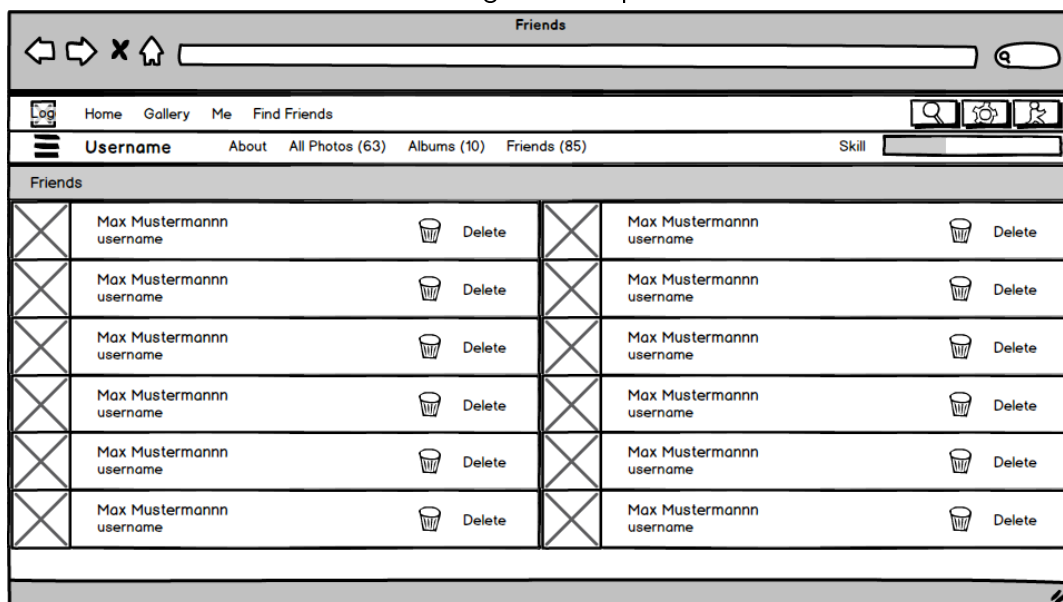


Abbildung 6: Mockup: Meine Freunde - Seite

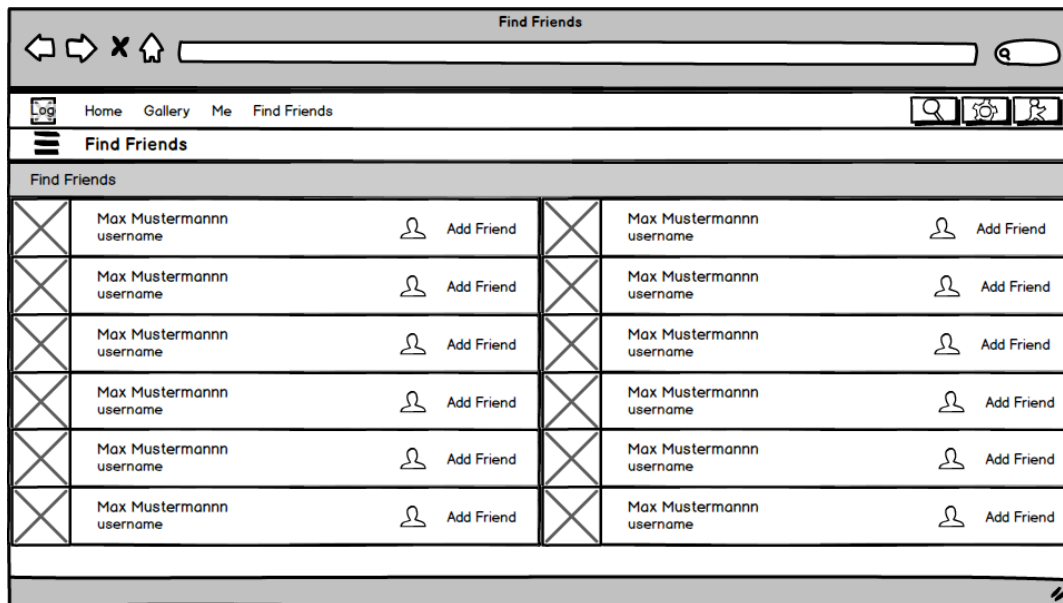


Abbildung 7: Mockup: Suche nach Freunde

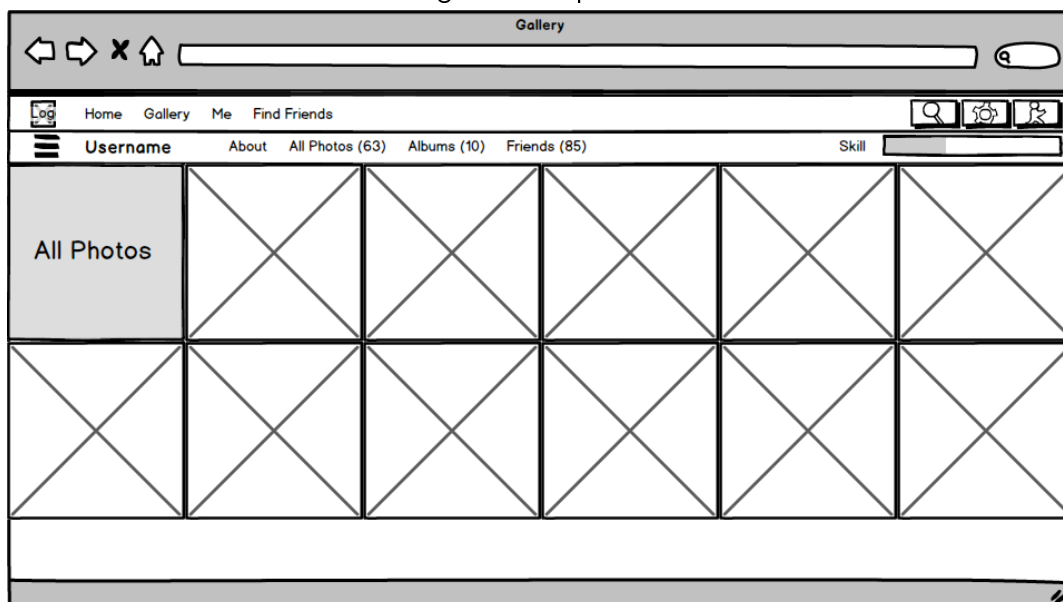


Abbildung 8: Mockup: Galerieansicht



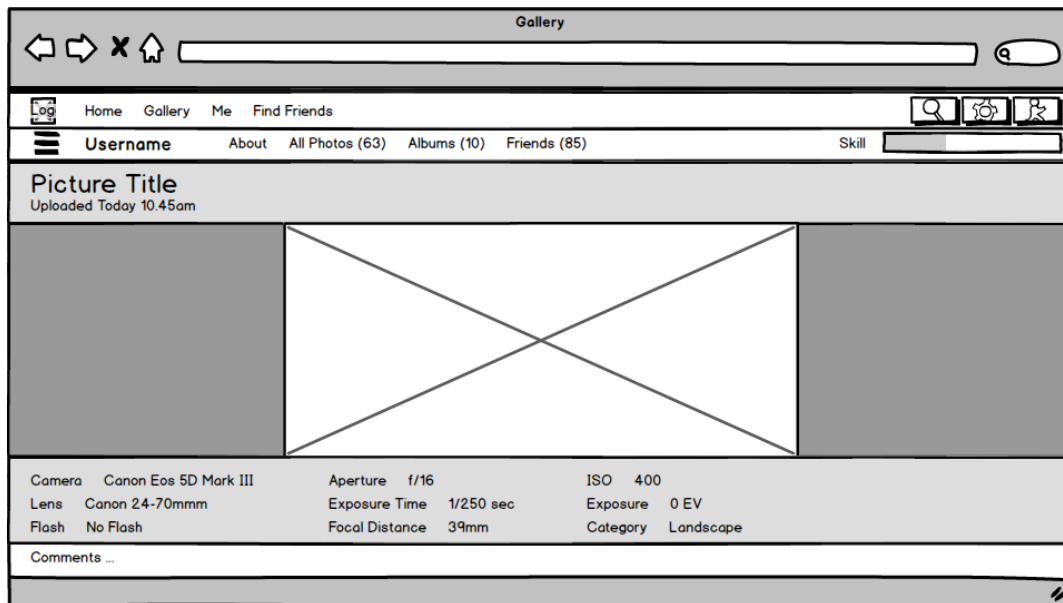


Abbildung 9: Mockup: Foto-Ansicht

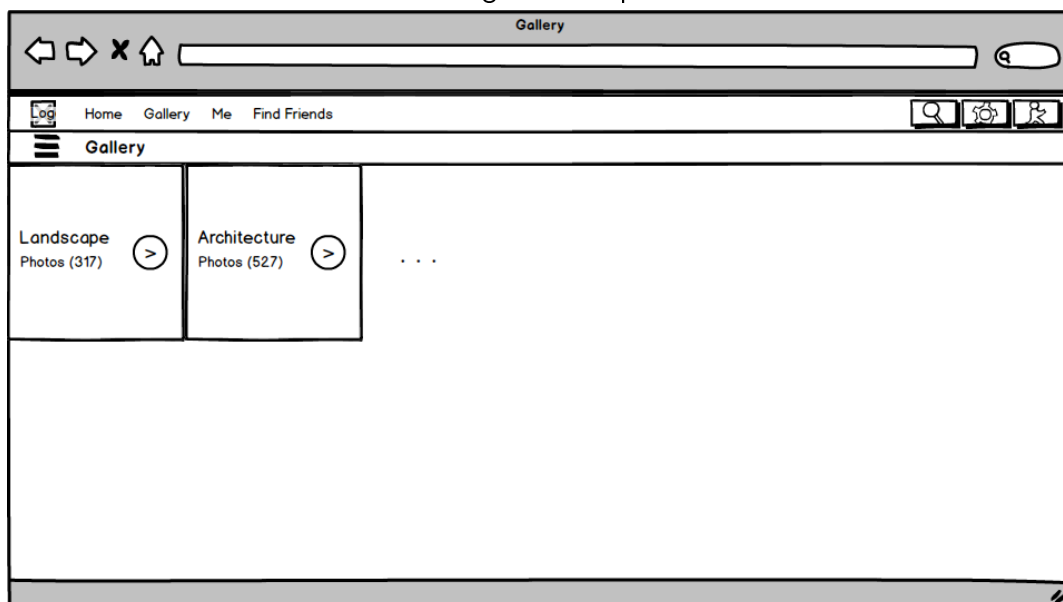


Abbildung 10: Mockup: Galerie Kategorie

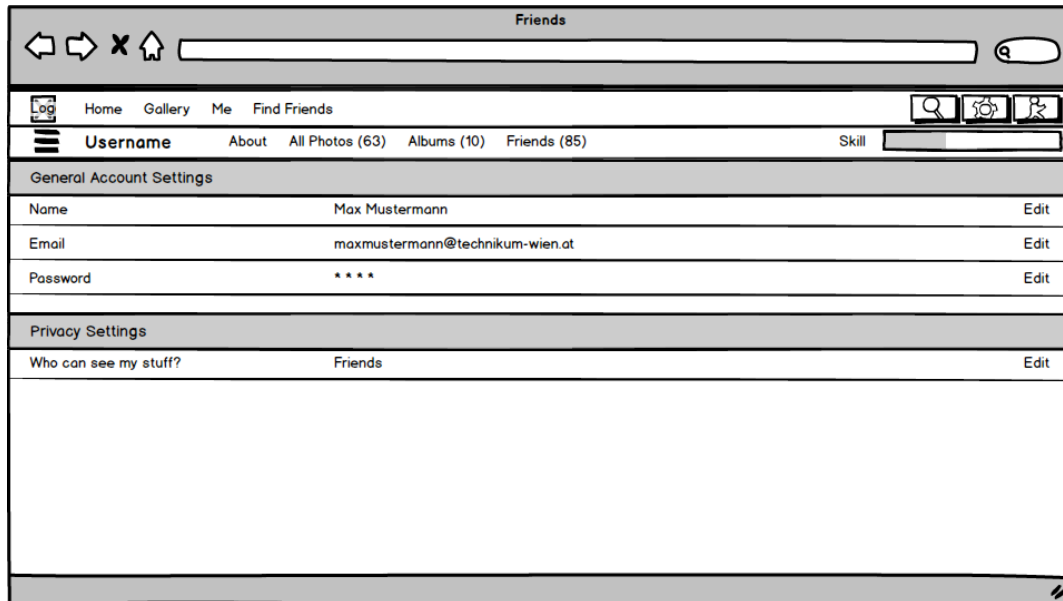


Abbildung 11: Mockup: Einstellungen

## 4.2 Systemschnittstellen

### 4.2.1 Webseite

### 4.2.2 Webserver

Der Webserver kommuniziert über Expressjs mit dem Client. Zuerst wird der Port angebunden und dann werden alle Requests an Expressjs weitergeleitet. Expressjs sendet dann den Verarbeiteten Request an das Routingmodul. Das Routingmodul hat wiederum eine Anbindung mit dem DB-Modul. Das DB-Modul kapselt dann schließlich die Datenbank zugriffe.

Expressjs selber hat zusätzlich noch einiges an Middleware die hier kurz zusammengefasst werden soll.

**connect-multiparty** Dient dazu Form-Requests die mit dem *enctype* 'multipart/form-data' gesendet werden, auch erfolgreich zu verarbeiten. Im wensentlichen wird der gesendet Body verarbeitet und in einem JSON Objekt gekapselt über den Request gesendet.

**body-parser** Dieses Middleware dient der Vorverarbeitung von Request-Bodies. Dazu wird der gesendete Body des Clients in ein JSON Objekt gekapselt. Um jedoch Binärdaten übertragen zu können, muss das zuvor erwähnte connect-multiparty Middleware dazu installiert werden.

**cookie-parser** Dieses Middleware dient dazu die vom Client gesendeten Cookies in ein einfach zu lesendes Array zu laden.

**cookie-session** Dieses Middleware erlaubt es Signed-Cookies zu übertragen und zu interpretieren.

**express-session** Einfach gesagt erlaubt dieses Middleware die Manipulation von der Client-Session. Hierfür wird eine eigene Sessionvariable angelegt die man beliebig ändern kann.

**Session** Es werden zum Teil wichtige Variablen in die Session gespeichert. Lesen kann man sie durch den Request mit der folgenden Syntax `'request.session.<variable>'`. Nachfolgend werden die existierenden Variablen samt Bedeutung beschrieben:

Variable	Bedeutung
username	Speichert den Anwendernamen.
series	Speichert die derzeitige Series.
token	Speichert den derzeitigen Token.
remember	Speichert ob der Anwender eingeloggt bleiben soll.

## Schnittstellen

Nodejs	⇔	Expressjs
Expressjs	⇔	Middleware
Expressjs	⇔	Routing
Routing	⇔	DB
DB	⇔	Datenbank

## 4.2.3 Datenbank

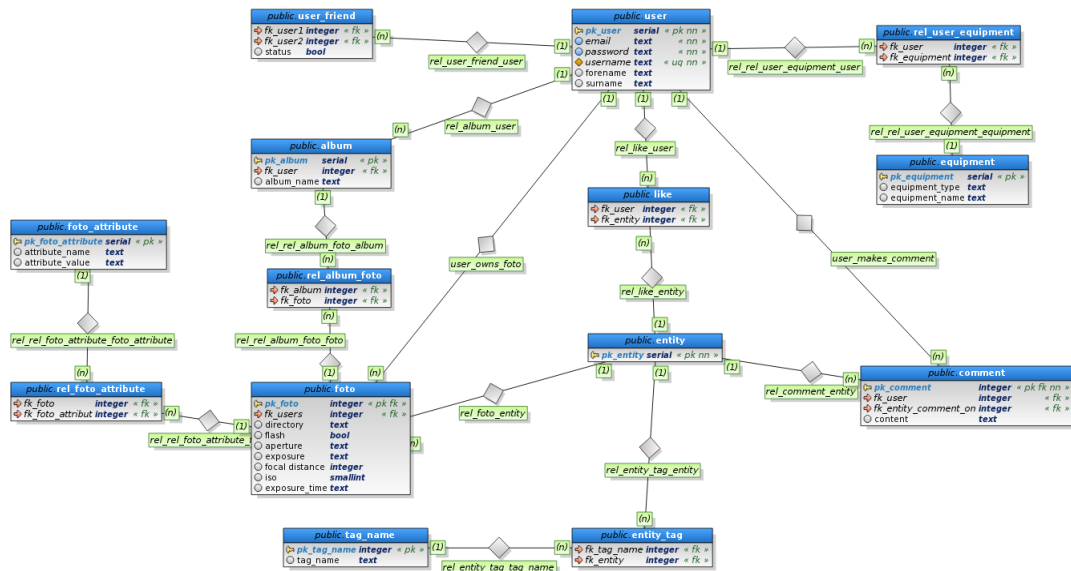


Abbildung 12: Datenbank: ERD

## 5 Nicht-Funktionale Anforderungen

### 5.1 Vorgaben zu Hardware und Software

#### 5.1.1 Software

- OpenSSL  $\geq$  1.0.1g
- Postgres  $\geq$  9.3
- Node.js  $\geq$  0.10.27
- ImageMagick  $\geq$  6.8.9-2
- PostgreSQL
- Ein moderner Browser welcher HTML5 & CSS3 unterstützt

Die benötigten Module von Nodejs werden durch die *package.json* definiert. Für eine Installation reicht das Kommando *npm install* in dem Verzeichnis mit der *package.json*.

#### 5.1.2 IP Forwarding

Es gibt zwei Server, einen HTTP-Server und einen HTTPS-Server. Der HTTP-Server bindet sich an den Port 8080, von daher ist es notwendig sämtlichen IP-Traffic vom Port 80 zum Port 8080 weiterzuleiten.

Der HTTP-Server dient der Weiterleitung von HTTP-Requests zu HTTPS-Requests. Der eigentliche Datenverkehr findet mit HTTPS statt. Hierfür muss dann der IP-Traffic von Port 443 auf den Port 43443 weitergeleitet werden.

### 5.2 Usability

## 6 Rahmenbedingungen

### 6.1 Meilensteine

- 1. Meilenstein: Login, Registrierung, Passwort vergessen fertigstellen (Frontend & Backend), 10 April 2014

### 6.2 Übersicht

- Entwicklungsumgebung Windows 8/Linux/OS X
- Javascript
- HTML5 & CSS3
- Node.js
- PostgreSQL

## 7 Lieferumfang

- Dokumentation
- Webseite

## **Abbildungsverzeichnis**