# Dokumentation MY PDMS

## Bedienungsanleitung

### Login Screen

### 

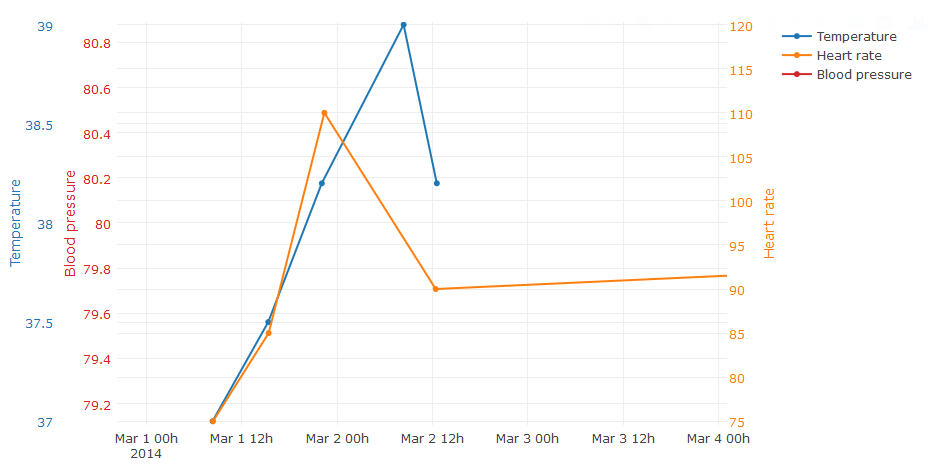
Dieser Screen dient dazu, die jeweiligen Benutzer zu authentifizieren. Nachdem der Benutzer seine Authentifizierungsdaten eingegeben hat, kann er mit der Taste „Enter“ oder mit einem Klick auf den Button „Login“ einloggen.

### Patientennavigation



Nach dem Login, wird der Benutzer auf seiner Homepage empfangen. Der Benutzer kann nun auf der linken Seite einen, der ihm zugewiesenen Patienten, auswählen.

### Vitaldaten Graph



Die erfassten Parameter „Temperatur“, „Blutdruck“ und „Puls“ werden dem Benutzer in einem Graphen präsentiert. Dieser kann wie folgt gelenkt werden:

* Mit einem Klick auf die jeweilige farbige Linie oben rechts, können die jeweiligen Parameter im Graphen ein- oder ausgeblendet werden.
* Die jeweiligen y- Achsen der Parameter können individuell mit der Maus nach oben und untern verschoben werden.
* Die x- Achse kann ebenfalls mit der Maus verschoben werden.

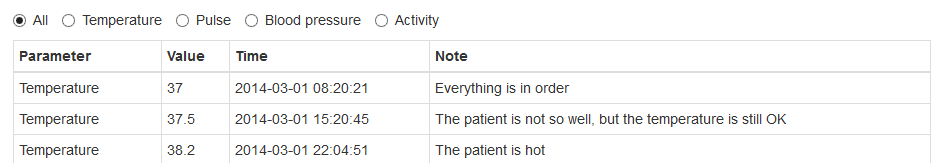
1 2 3 4 5 6



Wenn die Maus in die obere rechte Ecke bewegt wird, erscheint die Werkzeugleiste. Die Werkzeugleiste bietet dem Benutzer nochmals erweiterte Steuerungsmöglichkeiten für den Graphen.

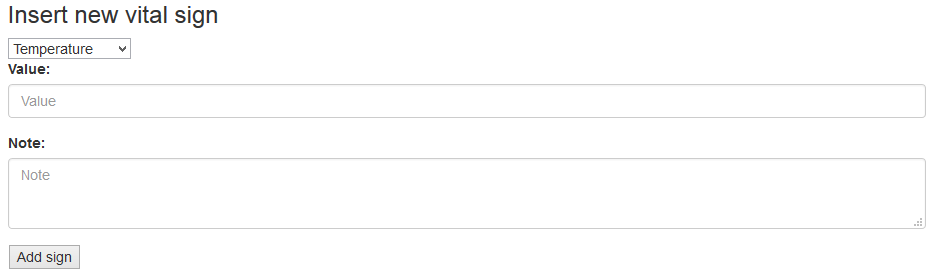
1. Speichern des Graphen als PNG- Bilddatei.
2. Eine erweiterte Zoomfunktion in horizontaler- und vertikaler Richtung, sowie in einem selektierten Viereck.
3. Ein Greifer, womit der Graph mithilfe der Maus beliebig verschoben werden kann.
4. Zwei +/- Buttons für einfaches Zoomen.
5. Eine automatische Skalierungsfunktion.
6. Ein Button um die Achsen in die Ausgangsposition zurückzusetzen.

### Vitalzeichen anzeigen



In der Parameterliste werden die erfassten Parameter zum Patienten angezeigt. Grundsätzlich werden immer alle Parameter angezeigt, dies soll verhindern, dass wichtige Parameter aufgrund eines Filters übersehen werden. Mithilfe der Radio-Buttons kann der Benutzer die Ansicht jedoch temporär einschränken. Der Filter wird zurückgesetzt, wenn die Seite neu geladen wird.

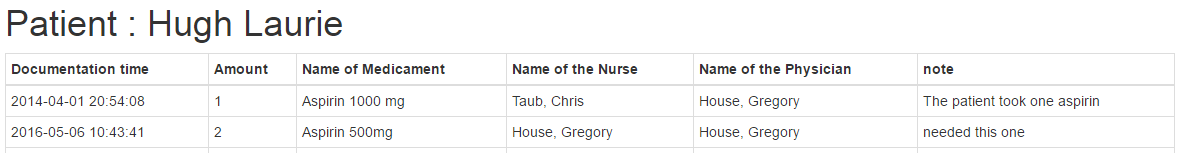
### Neues Vitalzeichen erfassen.



Ein neues Vitalzeichen kann mit dem Formular am Ende der Patientenansicht erfasst werden.

### Medikamentenabgaben anzeigen

Über die Vitalzeichenseite kann über „Show Medication“ die Medikationsseite geöffnet werden. Es werden die erfolgten Medikamentenabgaben dokumentiert, wobei der Name des Medikaments, die Menge, der Dokumentationszeitpunkt sowie der Name der Pflegenden und des Arztes angezeigt werden. Zusätzlich kann für jede Abgabe eine Notiz erfasst werden. Die Medikamente werden nach Dokumentationszeit aufsteigend sortiert.



### Medikamentenabgabe hinzufügen

Auf derselben Seite können neue Medikamentenabgaben hinzugefügt werden. Dazu können per Dropdown das Medikament und dessen Dosis, der Arzt und die Pflege ausgewählt werden. Zusätzlich kann die Menge und die Notiz in Textfeldern erfasst werden.



## Technische Dokumentation

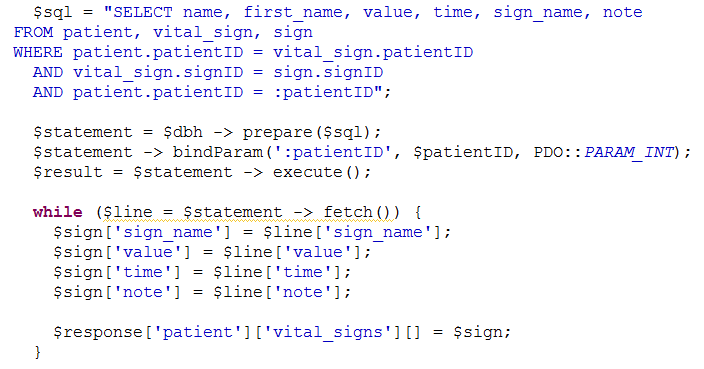
### Architektur

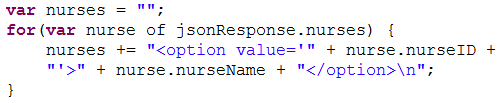
Die Applikation ist im MVC Pattern aufgebaut, wobei das Model in PHP, die View in HTML und der Controller in Javascript implementiert sind. Zusätzlich ist die Applikation als Drei-Schicht-Applikation mit einer beliebigen Anzahl Clients (Zugang per Webbrowser wie Chrome), einem Webserver (PHP) und einem Datenbankserver (MySQL) aufgebaut.

### Datenzugriff

Die für den Datenzugriff benötigten Informationen (Login, Passwort, URL des Datenbankservers) befinden sich im File pdo.inc.php.

Die Daten (z.B. Vital Signs oder Medikationsabgaben) werden in der MySQL-Datenbank gespeichert und in PHP mit Prepared Statements ausgelesen, insofern es sich um Statements mit Variablen handelt. Alle Daten, welche für eine Seite benötigt sind, werden in ein mehrdimensionales assoziatives Array ($response) gespeichert, welches anschliessend mit der PHP-Funktion json\_encode in einen JSON-String umgewandelt werden.



Diese Daten werden auf jeder Seite per AJAX-Aufruf vom Javascript-Controller geladen. Mittels JSON.parse werden anhand des JSON-String Javascript-Objekte erzeugt, welche anschliessend für die Darstellung in HTML aufbereitet werden. Hier das Beispiel für die Aufbereitung der Pflegenden, welche für die Abgabe der Medikamente hinterlegt werden kann. Der String „nurses“ wird anschliessend im entsprechenden Select hinterlegt.

### Patienten und Personen

Aktuell können Patienten und Personen (Arzt, Pflege, Admin) nur über die Datenbank erfasst und geändert werden. Sind sie korrekt erfasst erscheinen sie sofort in der Patientenauswahl bzw. in den Personenauswahlfeldern.

### Darstellung der Vitalwerte

Nebst der einfachen Darstellung der Vitalwerte in einer Tabelle werden diese auch als Graph dargestellt. Zur Visualisierung wurde die „plotly.js“ API verwendet. Die API erlaubt es dem Benutzer die maximale Freiheit bei der Betrachtung der Vitalwerte in die Hand zu geben.

Zur Zeit werden jeweils alle Daten im Graphen angezeigt. Es wäre sinnvoll dies künftig auf einen vernünftigen Zeitraum zu beschränken, sobald regelmässig Daten erfasst werden.

Die Konfiguration von plotly.js wird mithilfe von JSON Objekte vorgenommen. Um einen Line Chart darzustellen verlangt plotly.js pro Linie zwei Arrays mit Werten, einen für die x- Achse und einen für die y- Achse.

**var** chartData = {

temperature: {

time: new Array(),

value: new Array()

},

pulse: {

time: new Array(),

value: new Array()

},

blood: {

time: new Array(),

value: new Array()

}

};

Die Daten, welche für die Arrays benötigt werden, befinden sich in „jsonResponse.patient.vital\_signs“. Um die Daten in die Arrays zu schreiben, kann die gleiche Schleife verwendet werden, wie für die Darstellung der Vitalwerte in der Tabelle.

**for**(**var** sign of jsonResponse.patient.vital\_signs) {

…

…

**switch**(sign.sign\_name) {

**case** "Temperature":

chartData.temperature.time.push(sign.time);

chartData.temperature.value.push(sign.value);

**break**;

**case** "Pulse":

chartData.pulse.time.push(sign.time);

chartData.pulse.value.push(sign.value);

**break**;

**case** "Blood pressure":

chartData.blood.time.push(sign.time);

chartData.blood.value.push(sign.value);

**break**;

}

}

Die fertigen JSON Objekte werden dann den JSON Objekte, welche die Linien des Graphen konfigurieren.

**var** temperatureTrace = {

x: chartData.temperature.time,

y: chartData.temperature.value,

name: 'Temperature',

type: 'scatter',

marker: {

color: '#1f77b4',

line: {

color: '#1f77b4',

}

},

};

**var** pulsTrace = {

x: chartData.pulse.time,

y: chartData.pulse.value,

name: 'Heart rate',

yaxis: 'y2',

type: 'scatter',

marker: {

color: '#ff7f0e',

line: {

color: '#ff7f0e',

}

},

};

**var** bloodTrace = {

x: chartData.blood.time,

y: chartData.blood.value,

name: 'Blood pressure',

yaxis: 'y3',

type: 'scatter',

marker: {

color: '#d62728',

line: {

color: '#d62728',

}

},

};

Alle drei JSON Objekte werden dann in einem Array gespeichert, dass später der „plotly.js“ API übergeben wird.

**var** data = [temperatureTrace, pulsTrace, bloodTrace];

Vor dem Aufruf der API muss jedoch auch noch das Layout als JSON Objekt definiert werden. Hier kann die Beschriftung der Achsen und vieles mehr festgelegt werden.

**var** layout = {

margin: {

l: 50,

r: 50,

b: 35,

t: 0,

pad: 4

},

xaxis: {domain: [0.09, 0.95]},

yaxis: {

title: 'Temperature',

titlefont: {color: '#1f77b4'},

tickfont: {color: '#1f77b4'},

position: 0.0

},

yaxis2: {

title: 'Heart rate',

titlefont: {color: '#ff7f0e'},

tickfont: {color: '#ff7f0e'},

anchor: 'free',

overlaying: 'y',

side: 'right',

position: 0.95

},

yaxis3: {

title: 'Blood pressure',

titlefont: {color: '#d62728'},

tickfont: {color: '#d62728'},

anchor: 'x',

overlaying: 'y',

side: 'left',

},

};

Die Daten und die Konfiguration des Layouts, werden dann gemeinsam der API übergeben.

Plotly.newPlot('dataGraph', data, layout);

Der String „dataGraph“ entspricht der Id des Containers, indem wir den Graphen darstellen möchten.

<div id**=**"dataGraph"></div>

Weiteres zur „plotly.js“ API kann unter folgendem Link nachgelesen werden:

* <https://plot.ly/javascript/>

### CSS/Page Design

Beim Gestalten der Website haben wir auf „Bootstrap“ zurückgegriffen. Dadurch können wir mit einfachen Mittel eine ausgezeichnete Cross-Browser Kompatibilität gewährleisten.

Weiteres zu Bootstrap kann unter folgendem Link nachgelesen werden:

* <http://getbootstrap.com/>

### jQuery, Bootstrap.js, etc.

Auf die Verwendung von jQuery, und API’s die jQuery im Programmcode notwendig machen, wurde bei der Entwicklung bewusst verzichtet, da dies nicht dem Sinn der Übung entsprach. Dennoch wurden die entsprechenden API‘s für eine spätere Verwendung bereits in das Projekt miteingebunden.