

Grid-Layout mit smartVISU

Erläuterung des Prinzips

Dieses Grid-Layout wurde speziell für Smartphones und kleine Touchscreens, wie sie z.B für den Raspberry Pi verfügbar sind, entworfen. Das Grid-Layout arbeitet mit einem festen Raster und quadratischen Widgets. In der Standardkonfiguration besteht dieses Raster aus einer 8 x 4 Anordnung welche für das Querformat optimiert ist (es muss am Anfang eines Projekts entschieden werden, ob das Layout für Hoch- oder Querformat gestaltet werden soll).

Die folgende Abbildung 1 zeigt das Standard-Raster mit den zugehörigen Positionsangaben:

1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8

Abbildung 1: x,y-Positionen im Standardraster

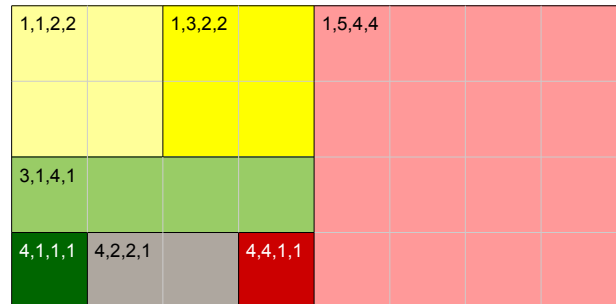


Abbildung 2: Beispiel Positionen und Größen

Die Position eines Widgets bestimmt die linke obere Ecke, anschließend folgt die Ausdehnung in x und y-Richtung. Folgende Positionen und Größen sind in der Abbildung 2 zu sehen, in Klammern stehen die verwendeten Widgets der Beispielseite (Abbildung 3):

- Helles gelb: Position 1,1 und die Größe 2 x 2 (dynamic)
- Dunkles gelb: Position 1,3 und die Größe ist ebenfalls 2 x 2 (shutter)
- Helles grün: Position 1,3 und eine Größe von 4 x 1 (value)
- Dunkles grün: Position 4,1 und eine Größe von 1 x 1 (rgb)
- Grau: Position, 4,2 mit einer Größe von 2 x 1 (pagelink)
- Dunkles rot: Position 4,4 und einer Größe von 1 x 1 (scene)
- Helles rot: Position 1,5 und eine Größe von 4 x 4 (camera)

Die Abbildung 3 zeigt beispielhaft Widgets mit der angegebenen Anordnung auf einem LG Nexus 5 mit Android 5.1.

Mit diesem Layout-Modell wird eine Beispielimplementierung mitgeliefert aus der die Verwendung deutlich werden sollte.

Zur Bearbeitung der Quelltexte muss ein geeigneter Editor eingesetzt werden, dies kann z. B. unter

Microsoft Windows der kostenlose Texteditor „Notepad++“ sein.

Für einen einfacheren Einstieg wird der Aufbau der angesprochenen Beispielseite im weitere Verlauf erläutert.

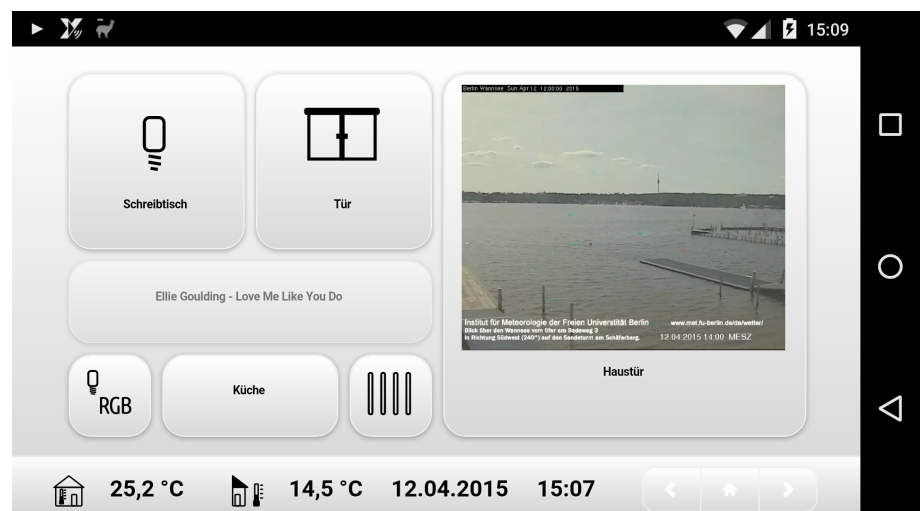


Abbildung 3: Die Anordnung mit Beispiel-Widgets

Vorüberlegungen

Das Grid-Layout ist als Ergänzung zu einem Standard-SmartVISU-Layout gedacht, mit einer schon angesprochenen speziellen Ausrichtung auf kleine Touchscreens.

Um das Grid-Layout parallel mit dem Standard-Layout zu Nutzen wird empfohlen das Projekt in einem Unterordner der eigenen smartVISU-Seite (pages/eigene_Seite/mobile) anzulegen. Bei dem vorliegenden Projekt wurde dieser Unterordner „mobile“ genannt und der Name sollte für den Einstieg auch nicht verändert werden, da er an vielen Stellen vorgegeben ist.

Jede Seite im Grid-Layout bekommt eine einzelne HTML-Datei. Um die Übersicht zu behalten sollten Unterordner verwendet werden, im Beispiel sind dies „eg“ und „dg“. Auch eine tiefere Verschachtelung ist denkbar, z. B. für jeden Raum einen extra Ordner (z.B. eg/kueche, eg/buero usw.).

Aufbau der Seiten

Alle Seiten sind mit einem einheitlichen Grundschema aufgebaut. Das Grundgerüst enthält immer die Zeilen wie in Text 1 dargestellt und kann als Basis für jede neue Seite kopiert werden.

```
1  /** Change word mobile to the name of your folder */
2  {% extends "mobile/base/base.html" %}
3  {% import "mobile/widgets/grid.html" as grid %}
4
5  {% block content %}
6      <div class="html disable_scroll" data-title="Beispielseite">
7          <div class="container">
8              <!-- ('id', 'infotext', row, column, size x, size y, ... )-->
9              Hier kommen später die einzelnen Widgets hin
10         </div>
11     </div>
12 {% endblock %}
13
14 {% block footer %}
15     {{ grid.footer ('footer_bsp', 'item.temp.innen', 'item.temp.aussen', '',
16         'index.php?page=mobile/haus', 'index.php?page=mobile/eg/kueche',
17         'visu.time.icon') }}
18 {% endblock %}
```

Text 1: Grundstruktur einer Seite mit dem Grid-Layout

Erläuterung:

In Zeile 2 und 3 werden die notwendigen Widgets eingebunden, mit „as grid“ wird festgelegt, dass die zur Verfügung stehenden Widgets mit dem Präfix „grid.“ eingebunden werden.

Die Zeilen 5 bis 12 definieren den eigentlichen Inhalt.

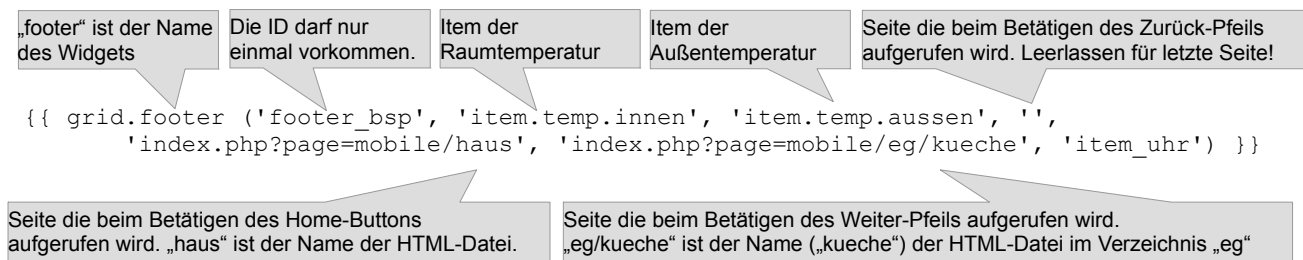
In Zeile 6 sollte der „data-title“ von „Beispielseite“ auf einen geeigneten Namen, der den Inhalt gut beschreibt, geändert werden, z. B. Küche oder Wohnbereich.

Ab Zeile 9 werden die Widgets nacheinander eingefügt, für jedes eine neue Zeile.

In Zeile 14 bis 16 wird die Fußzeile (footer) definiert. Die Fußzeile (Zeile 15) dient der Navigation und enthält grundlegende Informationen, im Einzelnen enthält das Widget folgende Elemente:

- Raumtemperatur
- Außentemperatur
- Datum und Uhrzeit
- Navigation bestehend aus Zurück, „Startseite“, nächste Seite und
- eine Angabe für ein Item welches die aktuelle Uhrzeit in Minuten enthält um eine analoge Uhr statt der digitalen zu verwenden.

Das footer-Widget ist etwas anders strukturiert als die übrigen und kann die folgenden Attribute annehmen:



Einfügen der Widgets im Inhaltsbereich

Im Verzeichnis mobile/widgets ist die Datei grid.html gespeichert. In dieser sind alle Verfügbaren Widgets enthalten und die notwendigen und optionalen Parameter kommentiert (erkennbar an den vorangestellten Sternen). Die Widgets arbeiten grundsätzlich mit svg-Icons wobei nur der Name ohne Dateipfad angegeben wird, z. B. light_light.svg.

Bei der Erstellung dieser Einführung standen folgende Widgets zur Verfügung:

- **grid.element**
Ein beliebiger Inhalt der im Parameter „content“ steht.
- **grid.pagelink**
Ruft eine andere Seite auf, als Link muss „index.php?page=mobile/...“ vorweg angegeben werden, dann folgt der Name der aufzurufenden HTML-Seite ohne die Dateierweiterung „.html“. Zusätzlich kann ein svg-Icon und Text angegeben werden.
- **grid.dynamic_pagelink**
Ähnlich wie das vorhergehende pagelink-Widget, verwendet als Icon aber ein dynamisches (siehe [smartvisu-Doku](#)), um einen Status zu visualisieren. Zusätzlich ist daher ein Item für den Schaltstatus und für den Wert erforderlich.
- **grid.weather_pagelink**
Spezielles Widget um die Wettervorhersage aufzurufen. Es kann ein Wetter-Icon und zusätzlich die (Vorhergesagte-)Temperatur anzeigen. Letzteres wird automatisch angezeigt wenn kein Text (txt) angegeben wurde.
- **grid.shutter**
Widget um Rollläden zu bewegen, mit einer Anzeige des Status (Fenster und Rollläden)
- **grid.dynamic**
Das Widget ermöglicht das Schalten und verändern von Werten, z.B. zum Schalten und Dimmen von Lampen.
- **grid.button**
Sendet bei der Betätigung einen definierten Wert, wenn kein Wert angegeben wurde wird eine „1“ gesendet.
- **grid.png_button**
Wie Button aber verwendet eine png-Grafikdatei als Icon.
- **grid.dual**
Der Button kann zwei Werte (z. B. „Ein“ und „Aus“) senden. Zusätzlich können unterschiedliche Icon und Texte den aktuellen Zustand darstellen.
- **grid.rgb**
Das Widget dient zum Verändern der Farbe von RGB-Leuchtmitteln.
- **grid.colordisc**
Wie „grid.rgb“ nur zur Auswahl der Farbe wird ein Farbkreis verwendet.
- **grid.symbol**
Das symbol-Widget dient nur zur Darstellung von Zuständen und verändert keine Werte.

- **grid.number**
Die Anzeige einer Zahl ohne Möglichkeit einen Wert zu verändern.
- **grid.value**
Anzeigen eines Wertes, z. B. eines Textes.
- **grid.float_popup**
Anzeige eines Buttons mit einer Fließkommazahl und eines Diagramms nach dem Auswählen.
- **grid.camera**
Anzeige eines Kamerabildes. Damit etwas zu erkennen ist, sollte das Widget entsprechend groß eingebunden werden.
- **grid.scene**
Ermöglicht die Auswahl einer Szene.
- **grid.plot_period**
Stellt in dem Widget einen Plot dar, z. B. mit einer Temperaturentwicklung, auch dieses Widget muss für eine fehlerfrei Darstellung entsprechend groß (beim Standardraster mindestens über zwei Reihen) eingebunden werden.
- **grid.weather_current**
Stellt in einem Widget das aktuelle Wetter dar, dieses Widget ist nur für hochkannt Widgets ausgelegt wie auf der Seite „wetter.html“ zusehen.
- **grid.weather_forecast**
Ähnlich wie „weather_current“ mit der gleichen Darstellung, nur mit Vorhersageinformationen, die Tage können als Offset angegeben werden.
- **grid.footer**
Gehört nicht zu den Inhaltswidgets und erzeugt die Fußzeile.

Die Widgets haben den gleichen prinzipiellen Aufbau, sie beginnen mit zwei geschweiften Klammern, gefolgt von dem oben angegebenen Namen. Anschließend folgen, eingeschlossen in runde Klammern, die erforderlichen Parameter und abgeschlossen wird jedes Widget durch zwei geschweifte schließende Klammern.

Die ersten sechs Parameter sind bei allen grid-Widgets gleich:

„widget“ ist durch den Namen zuersetzen.

Die ID muss eindeutig sein und darf sich nicht wiederholen.

Der optionale Infotext wird als Popup-Info angezeigt und teilweise auch als Widgetbeschriftung.

Position X und Y geben die Position im zugrundeliegenden Raster an. Bezugspunkt ist immer das linke obere Feld.

```

{{ grid.widget ( 'ID', 'Infotext', 'Position X', 'Position Y',
                'Größe X', 'Größe Y', ... ) }}

```

Größe X und Y geben die Anzahl der Felder in X- und Y-Richtung an die das Widget verwenden soll.

Hier folgen weitere Parameter die vom jeweiligen Widget abhängig sind.

Wenn ein Parameter als Optional gekennzeichnet ist, dann können diese am Ende weggelassen oder zwischen drin durch zwei Hochkommata (") ersetzt werden. Das Präfix „grid.“ wurde in der Grundstruktur im Text 1 in Zeile 3 festgelegt.

Aufbau der Beispielseite

Die Beispielseite enthält sieben Widgets und die Fußzeile. Die Grundstruktur entspricht dem Quelltext auf Seite 2 und dort werden ab der Zeile 9 die einzelnen Widgets eingefügt. Um die Beispielseite im Browser aufzurufen muss die folgende URL benutzt werden:

Den Begriff „server“ durch die IP-Adresse oder den Namen des Webserver mit dem SmartVISU erreichbar ist ersetzen.

Der Name der HTML-Datei, wobei die Dateiendung „.html“ weggelassen wird.

<http://server/index.php?page=mobile/beispiel>

Das dynamic-Widget

Das dynamic Widget besteht aus zwei Element, zum Einen das eigentliche Widget mit dem dynamischen Status-Icon und zum anderen aus einem Popup welches zum schalten und verändern eines Wertes dient. Die benötigten Parameter sind in der Datei grid.html im Verzeichnis „widgets“ zu finden. Dort sind folgende Informationen zum dynamic-Widget zu finden:

```
/**
 * Dynamic
 *
 * @param unique id for this widget
 * @param title of the dimmer (optional)
 * @param icon row
 * @param icon col
 * @param size x
 * @param size y
 * @param the dynamic icon
 * @param a gad/item for switching
 * @param a gad/item for a variable value
 * @param the minimum value if the slider is moved to total left (optional, default 0)
 * @param the maximum value if the slider is moved to total right (optional, default 255)
 * @param step between two values (optional, default 5)
 * @param readonly mode (default false)
 */
{% macro dynamic (id, info, row, col, sizex, sizey, dyn_icon, gad_switch, gad_value, min,
max, step, read_only) %}
```

Text 2: Auszug aus der Datei "widgets/grid.html"

Das Widget für die Beispielseite realisiert die Ansteuerung einer dimmbaren Lampe. Mit den Informationen aus grid.html kann das dynamic -Widget in das HTML-Grundgerüst in Zeile 9 eingefügt werden:

```
{{ grid.dynamic ('dimmer1', 'Schreibtisch', 1, 1, 2, 2, 'light',
                  'eg.buero.beleuchtung.schreibtisch.schalten',
                  'eg.buero.beleuchtung.schreibtisch.dimmwert',
                  0, 100, 5 ) }}
```

Nach den zwei öffnenden geschweiften Klammern wird das Widget mit „grid.dynamic“ eingebunden. Die ID ist bei diesem Widget „dimmer1“. Wie schon geschrieben muss die ID eindeutig sein, d.h. ein zweites Widget darf nicht auch „dimmer1“ heißen.

„Schreibtisch“ ist die Popup-Info und gleichzeitig die Beschriftung für das Widget.

Das Widget hat seinen Ursprung in der linken oberen Ecke des Layouts und bekommt daher die Reihe- und Spaltenangabe „1, 1“, da es zwei mal zwei Kästchen groß ist folgen noch die x- und y-Größenangaben „2, 2“.

Das Widget soll eine Lampe steuern und daher wird das dynamische Icon „light“ verwendet¹. Die verfügbaren Icons können der [smartVISU-Dokumentation](#) entnommen werden.

Nach dem „Icon“-Parameter folgen die Items zum Schalten der Lampe und zum Dimmen (Zeile 2 und 3).

¹ Das dynamische Icon „light“ steht noch nicht in smartVISU Version 2.7 zur Verfügung, sondern kann nur mit der Trunk-Version verwendet werden.

Die letzten drei Zahlen definieren den Wertebereich des Widgets, hier von 0 bis 100 (für letzteres ist der Default-Wert 255) und die minimale Schrittweite mit der die Lampe gedimmt wird.

Zusätzlich gibt es noch einen Parameter der angibt, dass das Widget nur einen Wert anzeigen soll, aber kein Popup generiert.

Das shutter-Widget

Das shutter-Widget dient zum Steuern eines Rollladens und ist im Beispiel wie folgt aufgebaut:

```
{{ grid.shutter ('shutter1', 'Tür', 1, 3, 2, 2,
                'eg.kueche.rollladen.sued.fahren',
                'eg.kueche.rollladen.sued.stopp',
                'eg.kueche.rollladen.sued.position',
                'eg.kueche.rollladen.sued.beschatten',
                'eg.kueche.fenster.sued') }}
```

Die ersten zwei Parameter sind gleich zum dynamic-Widget, deshalb gehe ich im folgenden auf die Notwendigkeit einer eindeutigen ID und den Info-Textes nicht mehr ein.

Nach dem Infotext-Parameter muss die Position und Größe angegeben werden. Das Widget ist neben dem „dimmer“-Widget angeordnet und bekommt daher als Positionsangabe „1, 3“ und für die Größenangabe erneut „2, 2“.

Im Anschluss folgen nacheinander die Items für die Bewegung, den Stopp-Befehl (optional), eine Rückmeldung über die Position, das Item für die Beschattung und des Fensterstatus (offen, gekippt, geschlossen). Die letzten beiden Parameter sind auch optional.

Damit ist das shutter-Widget mit allen notwendigen Parametern ausgestattet und kann nach Zeile 9 in einer neuen Zeile in das Grundgerüst eingefügt werden.

Als nächstes wird das value-Widget eingebunden.

Das value-Widget

Das value-Widget wird im Beispiel zur Anzeige des aktuell gespielten Musikstücks genutzt und hat nur wenige Parameter:

```
{{ grid.value ('title', 'Titel', 3, 1, 4, 1,
              'eg.kueche.musik.titel', '', '', '') }}
```

Der Anfang ist wie bei den vorhergehenden Widgets nur an einer anderen Position. Die Rasterzeilen 1 und 2 sind bereits durch die zwei ersten Widgets belegt und daher beginnt dieses Widget in der dritten Zeile und der ersten Spalte „3, 1“. Die Breite des value-Widgets beträgt im Beispiel vier und die Höhe ein Kästchen (4,1). Danach folgt das Item für den Titel. Die letzten drei optionalen Parameter werden hier nicht genutzt und enthalten den Button-Text, dann folgt eine mögliche Einheit und zum Schluss ein svg-Icon.

Das rgb-Widget

Das rgb-Widget zum Auswählen der Farbe und Helligkeit von RGB-Leuchtmitteln und -Strips ist im Beispiel nur 1x1 groß und wird daher ohne Beschriftung angezeigt.

Den Aufbau zeigt der folgende Quellcode:

```
{{ grid.rgb ('rgb1', '', 4, 1, 1, 1,
    'eg.kueche.beleuchtung.rgb_ap.schalten', 'Ein', 'Aus',
    'eg.kueche.beleuchtung.rgb_ap.rot.dimmwert',
    'eg.kueche.beleuchtung.rgb_ap.gruen.dimmwert',
    'eg.kueche.beleuchtung.rgb_ap.blau.dimmwert',
    50, 150, 8, 10) }}
```

Nach den Positions- und Größenangaben folgt das Item zum Schalten der Leuchte, sowie die 'Ein' und 'Aus'-Beschriftung für den switch-Button im Popup. Anschließend folgen die Items für die drei Grundfarbwerte rot, grün und blau. Im Anschluss werden die minimalen und maximalen Werte für die Farben angegeben. Dies ist manchmal erforderlich, weil z. B. die üblichen RGB-Stripes in der Regel bei niedrigen Werten keine Veränderung mehr zeigen und dies kann durch die Angabe ausgeglichen werden. Der erste Parameter (hier '50') definiert den minimalen Wert, dann für maximal '150'. Dann folgen noch zwei Zahlen um den Aufbau des Frabwahlfeldes zu definieren (Abstufungen und Anzahl der Farbfelder).

Das pagelink-Widget

Das pagelink-Widget die zum Aufrufen von anderen Seiten. Neben den Grundparametern gibt es noch drei weitere. Zum Einen der Link für die aufzurufende Seite, dann folgt die Widget-Beschriftung (hier: Küche) und zum Abschluss ein mögliches svg-Icon. Das SVG-Icon sieht bei einem ein Feld hohen und zwei Feld breiten Widget nicht gut aus und wird daher weggelassen.

```
{{ grid.pagelink ('pagelink1', 'Küchenbereich', 4, 2, 2, 1,
    'index.php?page=mobile/eg/kueche', 'Küche', '') }}
```

Das scene-Widget

Das scene-Widget dient dem Aufruf von vordefinierten Szenen. Das Widget soll in die letzte Zeile und an die vierte Stelle (4, 4). Die Ausdehnung ist jeweils eins in x,y-Richtung (1, 1).

Im Anschluss folgt das Szenen-Item, das svg-Icon zur illustration und zwei Wertreihen. Die kommaseparierten Wertereihen werden durch eine eckige Klammer „[“, begonnen und endet mit der schließen Klammer „]“. Zunächst die Beschriftungen und anschließend werden die Werte die bei der Auswahl der jeweiligen Bezeichnung gesendet werden sollen.

Der folgende Quelltext zeigt das vollständige Widget aus der Beispielseite.

```
{{ grid.scene ('scene1', '', 4, 4, 1, 1, 'eg.szenen.kueche_essen_wohnen',
    'control_4.svg', ['Alles Aus', 'Kochen', 'Essen',
    'Fernsehen', 'Party'], [13, 11, 10, 9, 12]) }}
```

Das camera-Widget

Als letztes wird noch ein Kamerabild mit dem camera-Widget eingebunden. Um auf einem kleinen Bildschirm etwas zu erkennen, bekommt es die Gesamthöhe und eine Breite von vier zugewiesen (4, 4). Die Position im Grid-Layout ist somit die erste Zeile und die fünfte Spalte (4, 5).

Anschließend folgt die URL des Videostreams, dieser wird beim klicken auf das Widget aufgerufen und sollte einen externen Player öffnen. Als letztes muss noch die URL zu einem Bild oder mjpeg-

Stream der Kamera als Parameter an das Widget mitgegeben werden. Das Bild bzw. der mjpeg-Stream wird von dem Widget dargestellt. Die beiden URLs im folgenden Beispiel müssen noch auf die Adressen der eigenen Kamera angepasst werden.

```
{{ grid.camera ('camera1', 'Haustür', 1, 5, 4, 4,
                'rtsp://192.168.xxx.xxx:554/0',
                'http://192.168.xxx.xxx/goform/stream?cmd=get%26channel=4') }}
```

Abschluss

Jetzt ist die Beispiel-Seite fertig und viele weitere Widgets warten auf ihren Einsatz. Wie schon beschrieben kann die Konfiguration der Widgets dem Quellcode von grid.html entnommen werden.

Hier nochmal der fertige Quelltext der Beispielseite:

```
/** Change word mobile to the name of your folder */
{% extends "mobile/base/base.html" %}
{% import "mobile/widgets/grid.html" as grid %}
{% block content %}
    <div class="html_disable_scroll" data-title="Test">
        <div class="container">
            <!-- (id, 'infotext', row, column, size x, size y, ... )-->
            {{ grid.dynamic ('dimmer1', 'Schreibtisch', 1, 1, 2, 2, 'light',
                            'eg.buero.beleuchtung.schreibtisch.schalten',
                            'eg.buero.beleuchtung.schreibtisch.dimmwert', 0, 100, 5) }}
            {{ grid.shutter ('shutter1', 'Tür', 1, 3, 2, 2,
                            'eg.kueche.rollladen.sued.fahren',
                            'eg.kueche.rollladen.sued.stopp', 'eg.kueche.rollladen.sued.position',
                            'eg.kueche.rollladen.sued.beschatten', 'eg.kueche.fenster.sued') }}

            {{ grid.value ('title', 'Titel', 3, 1, 4, 1, 'eg.kueche.musik.titel',
                            '', '', '') }}

            {{ grid.rgb ('rgb1', '', 4, 1, 1, 1,
                            'eg.kueche.beleuchtung.rgb_ap.schalten', 'Ein', 'Aus',
                            'eg.kueche.beleuchtung.rgb_ap.rot.dimmwert',
                            'eg.kueche.beleuchtung.rgb_ap.gruen.dimmwert',
                            'eg.kueche.beleuchtung.rgb_ap.blau.dimmwert', 50, 150, 8, 10) }}
            {{ grid.pagelink ('pagelink1', 'Küchenbereich', 4, 2, 2, 1,
                            'index.php?page=mobile/eg/kueche', 'Küche', '') }}
            {{ grid.scene ('scenel', '', 4, 4, 1, 1, 'eg.szenen.kueche_essen_wohnen',
                            'control_4.svg', ['Alles Aus', 'Kochen', 'Essen', 'Fernsehen',
                            'Party'], [13, 11, 10, 9, 12]) }}

            {{ grid.camera ('camera1', 'Haustür', 1, 5, 4, 4,
                            'http://www.berlin.de/webcams/wannsee/webcam.jpg',
                            'http://www.berlin.de/webcams/wannsee/webcam.jpg') }}

        </div>
    </div>
{% endblock %}
{% block footer %}
    {{ grid.footer ('test_footer', 'eg.kueche.temperatur.luft',
                    'aussen.nordseite.temperatur', 'index.php?page=mobile/eg/index',
                    'index.php?page=mobile/haus', 'index.php?page=mobile/eg/kueche') }}
{% endblock %}
```


Weitere Hinweise und Informationen

- Das vorgegebene Raster ist so ausgelegt, dass es auf kleinen Bildschirmen im Querformat noch gut ablesbar und bedienbar ist. Die meisten Widgets sollten hierzu in der Größe 2x2 eingesetzt werden. Die Größe 1x1 ist nur für wenig verwendete Aktionen gedacht, die länglichen Widgets mit einer Höhe von eins dienen zur Darstellung von Texten oder Zahlen ohne Icon.
- Wenn das vorgegebene Raster trotzdem verändert werden soll, z. B. um den Platz von Tablets besser zu nutzen, dann sind die Anzahl der Reihen und Spalten, sowie die Abstände in der Datei „mobile/grid.js“ in den ersten Zeilen zu finden.
- Ein Mischen von unterschiedlichen Rasterlayouts ist nicht vorgesehen. Das Verzeichnis „mobile“ kann aber kopiert und umbenannt werden und darin dann mit einem anderen Raster gearbeitet werden. Allerdings gilt auch hier die Einschränkung, dass das „mobile“ dann in allen Dateien auf den neuen Verzeichnisnamen geändert werden muss.
- Die Basis sind nahezu bei allen Widgets SVG-Grafiken, die ohne Pfad angegeben werden.
- Die Datei „root.html“ und die Dateien im Verzeichnis „base“, erzeugen den grundsätzlichen Aufbau, im Verzeichnis „vendor“ liegen benötigte Skripte aus Fremdquellen und widget enthält die Logik und das Design.