ioBroker.vis im Material Design Style

Anleitung zur Material Design CSS

Grundsätzlich wird die Visualisierung mit den bekannten basic und jqui Controls entworfen. Diese erhalten lediglich CSS-Klassenzuweisungen um im Browser im Material Design Style gerendert zu werden. Es gibt nicht für alle Controls entsprechende CSS-Klassen.

Die CSS-Anweisungen müssen dem Projekt auf der CSS-Registerseite unter „Projekt" zugewiesen werden. Sie stehen dann sowohl zur Anzeige im Editor als auch zur Laufzeit zur Verfügung. Sie sind so gestaltet, dass sie sich im Editor nur auf das View-Editfenster auswirken und nicht auf den Rest des Editors. Dazu ist es auch notwendig, ein wenig JS Code auf der Skripte-Registerseite einzufügen (einfach hier mit Copy&Paste herauskopieren oder als Basis eines der Beispiel-Projekte verwenden).

Oder, noch einfacher, eines der am Ende genannten Demo-Projekte als Basis verwenden.

```
// Zur sicheren CSS-Erkennung der Runtime eine CSS-Klasse anlegen
document.documentElement.className +=  " mdui-runtime";

// Überprüfen ob touch zur Verfügung steht und entsprechend eine
// CSS Klasse touch bzw no-touch erzeugen
document.documentElement.className +=
    (("ontouchstart" in document.documentElement) ? " mdui-touch" : " mdui-
notouch");

// Binden des click() Events an den #vis_container mit der Delegation
// für die mdui-l/rbar Klassen. Ein direktes Binding funktioniert durch
// den view-Aufbau nicht, da dieses Script dann beim zurück-Navigieren
// nicht mehr aufgerufen wird.
setTimeout(function () {
    // click-Event für das left-nav Element zum Öffnen
    $("#vis_container").on( "click", ".mdui-lnavbutton", function() {
        $( ".mdui-lnav" ).addClass( "mdui-lnav-open" );
    } );
    // click-Event für die left-nav zum Schließen
    $("#vis_container").on( "click", ".mdui-lnav", function() {
        $( ".mdui-lnav" ).removeClass( "mdui-lnav-open" );
    } );

    // click-Event für das right-nav Element zum Öffnen
    $("#vis_container").on( "click", ".mdui-rnavbutton", function() {
        $( ".mdui-rnav" ).addClass( "mdui-rnav-open" );
    } );
    // click-Event für die right-nav zum Schließen
    $("#vis_container").on( "click", ".mdui-rnav", function() {
        $( ".mdui-rnav" ).removeClass( "mdui-rnav-open" );
    } );
}, 1000);
```

In der folgenden Beschreibung sind notwendige Einstellungen der basic- und jqui-Controls sind mit "MUSS" gekennzeichnet, optionale mit "KANN".

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	...	Der Wert MUSS gesetzt werden
	CSS Klasse	...	Optional

Farben

Als Vorgabe ist der Style insgesamt in schwarz-weiß gehalten, es besteht aber die Möglichkeit über CSS Angaben die Darstellung der Text, Buttons usw. auch farbig zu erhalten. Hierzu stehen spezielle mdui-(color) (Schriftfarben), mdui-(color)-bg (Hintergrundfarben) und mdui-(color)-acc (Akzentfarben) CSS Klassen zur Verfügung. Verwendet werden die durch das Material Design vorgegebenen Farben.

Als Farben (color) stehen zur Verfügung:

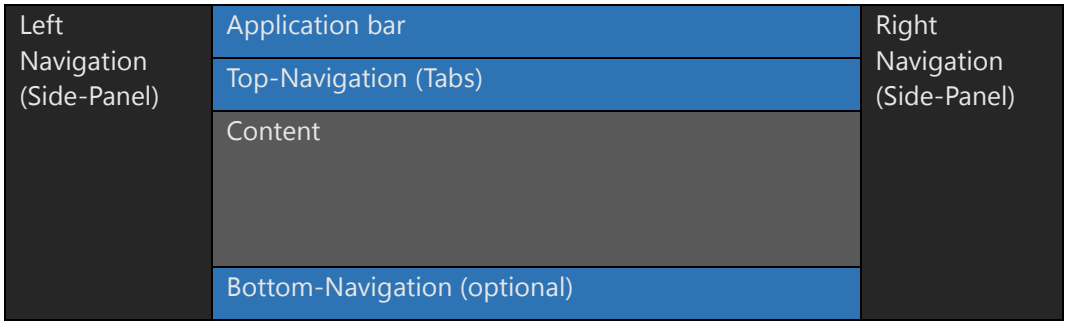
teal	**teal**	**indigo**	**indigo**	**amber**	**amber**
lime	**lime**	**red**	**red**	green	green
yellow	yellow	brown	brown		

TIPP
Farben ziehen die Aufmerksamkeit des Betrachters auf sich. Damit sie diese Aufgabe gut erledigen können, dürfen sie nicht in einer Vielzahl von anderen Farben untergehen. Das Grundlayout sollte also bewusst einfarbig gestaltet sein, damit die Signalfarben dann umso deutlicher auffallen.

Layout

Das normale Layout einer Seite (Page) besteht aus sechs Bereichen, wobei bis auf den Content-Bereich alle optional sind.

Aufbau einer Seite:



In der top-navigation werden die Buttons für die Navigation durch die einzelnen Seiten (Views) angezeigt. Je nach Navigationstiefe kann die top-navigation angepasst werden, in dem in den Seiten einfach andere top-navigations eingebunden werden. So besitzt z.B. die Hauptseite eine andere top-navigation als z.B. die Hausseite, in welcher die Räume aufgeführt werden. Views, die als top-navigation genutzt werden, sollten mit "tnav" beginnen, z.B. "tnavMain", "tnavHaus".

Im Content wird später die View mit dem eigentlichen Inhalt der Seite eingeblendet.

Die bottom-navigation kann nun entweder wie die application-bar oder wie die tab-navigation genutzt werden. Views, die als bottom-navigation genutzt werden, sollten mit "bnav" beginnen, z.B. "bnavMain". Bottom-navigations werden selten verwendet.


Die left-navigation stellt das Application-Menü dar, welches beim Betätigen der Menü-Schaltfläche von links eingeblendet wird. Hier können z.B. noch einmal die Links zu den einzelnen Seiten aufgeführt werden. Views, die als left-navigation genutzt werden, sollten mit "lnav" beginnen, z.B. "lnavMain". Normalerweise gibt es genau eine left-navigation.

 Die right-navigation stellt das Funktions-Menü dar, welches beim Betätigen der Funktion-Schaltfläche von rechts eingeblendet wird. Hier können z.B. kontextabhängige Funktionen aufgeführt werden. Views, die als right-navigation genutzt werden, sollten mit "rnav" beginnen, z.B. "rnavMain". Auf unterschiedlichen Seiten können verschiedene right-navigations verwendet werden.

Wichtig
Alle genannten Komponenten werden nicht direkt in den Seiten eingefügt, sondern sind jeweils eigene Views, die über das Widget ***basic-view in widget Container*** in die Seiten eingebunden werden! Näheres bei der jeweiligen Beschreibung.

Application bar

Application bar View

Hierbei handelt es sich um eine View, die später in den Application bar Containern verwendet wird. Es werden Buttons und Texte untergebracht, die permanent sichtbar sein sollen. Wie z.B. Menü- und Home-Buttons, Uhrzeit und Tagesanzeige. In der Regel hat ein Projekt genau eine application-bar, es sind aber auch mehrere möglich. Views, die als application-bar genutzt werden, sollten mit "abar" beginnen, z.B. "abarMain". Die Höhe sollte 48px betragen.

Die View sollte z.B. die Schaltflächen zum Öffnen der left- und right-navigation beinhalten, damit diese als solche erkannt werden, muss ihnen jeweils eine CSS-Klasse zugewiesen werden.

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-lnavbutton	

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-rnavbuttton	

Application bar Container

Auf jeder Seite, auf denen eine Application bar dargestellt werden soll, ist ein Widget ***basic-view in widget Container*** einzufügen, in welchem dann die Application bar-View angezeigt wird. Den Containern (nicht den Views!) auf den Seiten wird dann die zugehörige CSS Klasse zugewiesen, sie sollten alle eine Höhe von 48px haben.

Muss?	Eigenschaft	Werte	Beschreibung
Ja	Width Height	100% 48px	Kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 1280px

Ja	CSS Klasse	mdui-abar	
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe (siehe dort)

Top-Navigation, Bottom-Navigation

Top/Bottom-Navigation View

Hierbei handelt es sich um Views, die später in den Top/Bottom-Navigation Containern verwendet werden. Normalerweise werden hier Buttons untergebracht, welcher der Navigation dienen. Views, die als Top/Bottom-Navigation genutzt werden, sollten mit "tnav" bzw. „bnav“ beginnen, z.B. "tnavMain". Die Höhe sollte 48px betragen.

Die in den Views verwendeten **Buttons** sollten die folgenden Einstellungen erhalten:

Muss?	Eigenschaft	Werte	Beschreibung
	Width Height	auto 32px	
	CSS Klasse	mdui-flatbutton	Reine Textbuttons
	CSS Klasse	mdui-float mdui-float-right	Damit werden die Buttons automatisch links (rechts) angeordnet. Insbesondere wenn man im Button-Text mit Breitenbedingungen arbeitet, ist dieses sehr sinnvoll. Da das floaten in Abhängigkeit der Erstellungsreihenfolge passiert, muss man evtl. über Kopieren/Einfügen die Buttons in die richtige Reihenfolge bringen – ioBroker hat m.W. dafür keine eigene Funktion.
	Text	„Buttontext“	Wenn man hier Rücksicht auf ein responsive Design nehmen möchte, kann man den Text so gestalten, dass er bei Bildschirmauflösungen von max. 480px und mehr als 480px unterschiedlich dargestellt wird. Hierzu sind die CSS Klassen mdui-show480 und mdui-hide480 zu nutzen. Bsp: EGErdgeschoß
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe (siehe dort)
	CSS Klasse	mdui-(color)	Schriftfarbe

TIPP
Um Controls am rechten Rand so zu verankern, dass sie mit der Screen-Breite skalieren, kann man bei der "left" Angabe "calc(100% - nnnpx)" angeben, mit nnn = Breite des Controls.

Top/Bottom-Navigation Container

Auf jeder Seite, auf denen eine solche Navigation dargestellt werden soll, sind entsprechende Widget ***basic-view in widget Container*** einzufügen, in welchem dann die Top/Bottom-Navigation Views angezeigt werden. Den Containern (nicht den Views!) auf den Seiten wird dann die zugehörige CSS Klasse zugewiesen, sie sollten alle eine Höhe von 48px haben.

Top-Navigation:

Muss?	Eigenschaft	Werte	Beschreibung
	Width Height Top Left	100% 48px 48px 0px	Width kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 1280px
Ja	CSS Klasse	mdui-tnav	Top-Navigation
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe (siehe dort)
	CSS Klasse	mdui-(color)-acc	Akzentfarbe um den aktuellen Navigations-Button zu kennzeichnen

Bottom-Navigation:

Muss?	Eigenschaft	Werte	Beschreibung
	Width Height Top Left	100% 48px calc(100% - 48px) 0px	Width kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 1280px Top kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 672px
Ja	CSS Klasse	mdui-bnav	Top-Navigation
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe (siehe dort)
	CSS Klasse	mdui-(color)-acc	Akzentfarbe um den aktuellen Navigations-Button zu kennzeichnen

Content

Content View

Hierbei handelt es sich den View, der später in den Content Containern verwendet wird und den echten Inhalt der Seiten bekommt. Zu jedem Seiten-View gibt es also genau einen Content-View. Views, die als Content genutzt werden, sollten mit "cont" beginnen, z.B. "contMain".

In den Content Views kann man entweder die Widgets direkt hinein setzen, oder, wenn man ein responsive Design haben möchten, fügt man nur Widget ***basic-view in widget Container*** ein, die als Platzhalter für die (Card-) Views dienen, welche dann die Widgets enthalten. Dieses ist notwendig, da ansonsten kein automatischen „floaten" des Inhalts möglich ist.

Diesen Widget ***basic-view in widget Container***: können zugewiesen werden:

Muss?	Eigenschaft	Werte	Beschreibung
	Width	312 px	Beim responsive Design sollte man immer „mobile first" im Blick haben, also die Breite der einzelnen Contaoner so wählen, dass sie auch noch auf den gewünschten Endgeräten komplett dargestellt werden können. Bei heutigen Smartphones, die eine Auflösung von 720x1280px haben, wären das 360px (wegen der doppelten Pixeldichte).
	CSS Klasse	mdui-float mdui-float-right	Damit werden die Container automatisch links (rechts) angeordnet. Da das „floaten" in Abhängigkeit der Erstellungsreihenfolge passiert, muss man evtl. über Kopieren/Einfügen die Buttons in die richtige Reihenfolge bringen – ioBroker hat m.W. dafür keine eigene Funktion.
	CSS Klasse	mdui-card	Darstellung des Containers als „Card", also mit Schatten
	CSS Klasse	mdui-tile	Darstellung des Containers als „Tile"
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe (siehe dort)

Content Container

Auf jeder Seite gibt es genau ein Widget ***basic-view in widget Container*** welches den Content View darstellen wird.

Muss?	Eigenschaft	Werte	Beschreibung
	Width Height Top	100% calc(100% - 2 * 48px)	Width kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 1280px

	Left	96px 0px	Wenn mit Bottom-Navigation, dann ist Height: calc(100% - 3 * 48px)
Ja	CSS Klasse	mdui-content	
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe (siehe dort)

Left-Navigation, Right-Navigation

Left/Right-Navigation View

Hierbei handelt es sich um Views, die später in den Left/Right-Navigation Containern verwendet werden. Normalerweise werden hier Buttons untergebracht, welcher der internen und externen Navigation dienen. Views, die als Left/Right-Navigation genutzt werden, sollten mit "lnav" bzw. „rnav" beginnen, z.B. "lnavMain".

Muss?	Eigenschaft	Werte	Beschreibung
	Width	max 288px	Da die Left/Right-Navigation Container später mit einer Breite von 288px angezeigt werden, sollte diese im View eingehalten werden
	Height		Egal, da später im Left/Right-Navigation Container vertikal gescrollt wird
	Top Left	8px 8px	

Left/Right-Navigation Container

Auf jeder Seite, auf denen eine solche Navigation dargestellt werden soll, sind entsprechende Widget ***basic-view in widget Container*** einzufügen, in welchem dann die Left/Right-Navigation Views angezeigt werden. Den Containern (nicht den Views!) auf den Seiten wird dann die zugehörige CSS Klasse zugewiesen.

Left-Navigation Container

Muss?	Eigenschaft	Werte	Beschreibung
	Width Height Top Left		Egal, da diese Werte nur für den Designer gelten, zur Laufzeit werden die Container automatisch am linken Rand angezeigt. Man kann also auch eine (nicht störende) Größe von 48x48px verwenden.
Ja	CSS Klasse	mdui-lnav	Left-Navigation
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe (siehe dort)

Right-Navigation Container

Muss?	Eigenschaft	Werte	Beschreibung
	Width Height Top Left		Egal, da diese Werte nur für den Designer gelten, zur Laufzeit werden die Container automatisch am linken Rand angezeigt. Man kann also auch eine (nicht störende) Größe von 48x48px verwenden.
Ja	CSS Klasse	mdui-rnav	Left-Navigation
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe (siehe dort)

Cards

Cards können, müssen aber nicht verwendet werden. Mit Cards lassen sich Widgets optisch gruppieren. Für eine Card wird das Widget ***basis-html*** verwendet.

Im **responsive Design** wird die Card nicht dem View mit den anzuzeigenden Werten zugewiesen, sondern jeweils dem Container im Content, in welchem der View angezeigt wird.

Sollen Cards Titel und Untertitel haben, so können hierfür die unter „Labels" beschriebenen CSS-Styles verwendet werden.

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-card	
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein

Tiles

Tiles können, müssen aber nicht verwendet werden. Mit Cards lassen sich Widgets optisch gruppieren. Für eine Card wird das Widget ***basis-html*** verwendet. Gegenüber den Cards besitzen sie keinen Schatten.

Im **responsive Design** wird die Card nicht dem View mit den anzuzeigenden Werten zugewiesen, sondern jeweils dem Container im Content, in welchem der View angezeigt wird.

Sollen Cards Titel und Untertitel haben, so können hierfür die unter „Labels" beschriebenen CSS-Styles verwendet werden.

Muss?	Eigenschaft	Werte	Beschreibung

Ja	CSS Klasse	mdui-tile	
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein

Labels

Für ein Label (Text) wird das ***basis-html*** Widget verwendet. Es kann sowohl für die Darstellung von Titeln, Untertiteln, Labels als auch von Werten verwendet werden. Labels werden etwas dunkler und kleiner als Werte dargestellt.

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-label oder mdui-value oder mdui-title oder mdui-subtitle	Label-Text Wert Titel, Überschrift Untertitel
	CSS Klasse	mdui-(color)	Schriftfarbe. (color) kann eine der Farben (siehe dort) sein
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein

TIPP
Da bei jedem Attribut auch auf ioBroker Variable zugriffen werden kann, ist auch eine Farbsteuerung über Variable möglich. Hierzu eine Zeichenvariable deklarieren und ihr den gewünschten Farbwert "red", "green", ... zuweisen. In der CSS Klasse dann mdui-{meine Variable} verwenden.

Bsp: mdui-{javascript.0.farbwert-temperatur}

Buttons

Für einen Button können verschiedene ***jqui-Button*** Widgets verwendet werden. Bei den meisten funktioniert die Darstellung korrekt, wenn eine der Button CSS Klassen zugewiesen wird. Im Material Design gibt es drei Button Arten: flat-button, raised-button und floating-button.

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-flat-button oder mdui-raised-button oder mdui-floating-button	

	CSS Klasse	mdui-(color)	Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe für flat ist „blau“, für die anderen "weiß".
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "blau".

Radio Buttons

Für Radio-Eingaben wird eines der ***jqui-Radio...*** Widgets verwendet.

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-radio	
	CSS Klasse	mdui-(color)	Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß".
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "transparent".
	CSS Klasse	mdui-(color)-acc	Akzentfarbe um die aktuelle Auswahl zu kennzeichnen. Vorgabe ist „blau“.

Inputs

Für einen Input können verschiedene ***jqui-Input*** Widgets verwendet werden. Bei den meisten funktioniert die Darstellung korrekt. Material Design typisch werden Inputs nur durch eine untere Rahmenlinie angezeigt, die beim Fokus farbig wird.

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-input	
	CSS Klasse	mdui-(color)	Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß".
	CSS Klasse	mdui-(color)-bg	Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "transparent".

Switches

Für einen Switch wird das Widget ***basis-bool checkbox*** verwendet.

Muss?	Eigenschaft	Werte	Beschreibung

Ja	CSS Klasse	mdui-switch	
Ja	HTML anhängen	<label for="(id)_checkbox"> </label>	**Wichtig!** Wobei (id) durch die ID des Controls selbst ersetzt werden muss, Bsp: "w00033_checkbox"
	CSS Klasse	mdui-(color)-acc	Hiermit kann festgelegt werden, welche Farbe im EIN Zustand verwendet werden soll. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß".

Wichtig
Damit der Switch funktioniert, muss die oben angegebenen HTML-anhängen Erweiterung eingegeben werden.

Demo Projekte

Mit auf GitHub befinden sich drei Demo-Projekte, welche eine ioBroker.vis Visualisierung ähnlich einer Android-App mit Application-bar, Tab-Navigation und Bottom-Navigation bieten. Die Controls dieser Demo-Projekte sind nicht mit echten Instanzen verbunden. Die verwendeten Symbole (Icons) befinden sich alle im Projektunterordner „images“.

Die Demo-Projekte, also die ZIP-Datei, können in ioBroker.vis unter „Projekt importieren“ eingelesen werden.

MD_Demo.zip

Ein ioBroker Projekt in welchem alle Möglichkeiten des Material Design Styles dargestellt werden.

MD_Simple.zip

Ein ioBroker Projekt welches als Basis für ein eigenes Projekt verwendet werden kann. Es ist mit allen notwendigen Views für die Navigation ausgestattet. Die Seiten sind neutral gehalten.

MD_Complex.zip

Ein ioBroker Projekt welches in ähnlicher Form bei mir eingesetzt werden wird. Zwar niht ganz bis ins Detail ausgearbeitet, aber es zeigt sehr gut den vollen Funktionsumfang. U.a. auch mit mehreren Navigationsebenen (Haus-, Sicherheitsbereich).

Lizenz

The MIT License (MIT)

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Deutsche Übersetzung

Copyright (c) 2017ff Uhula

Hiermit wird unentgeltlich jeder Person, die eine Kopie der Software und der zugehörigen Dokumentationen (die "Software") erhält, die Erlaubnis erteilt, sie uneingeschränkt zu nutzen, inklusive und ohne Ausnahme mit dem Recht, sie zu verwenden, zu kopieren, zu verändern, zusammenzufügen, zu veröffentlichen, zu verbreiten, zu unterlizenzieren und/oder zu verkaufen, und Personen, denen diese Software überlassen wird, diese Rechte zu verschaffen, unter den folgenden Bedingungen:

Der obige Urheberrechtsvermerk und dieser Erlaubnisvermerk sind in allen Kopien oder Teilkopien der Software beizulegen.

DIE SOFTWARE WIRD OHNE JEDE AUSDRÜCKLICHE ODER IMPLIZIERTE GARANTIE BEREITGESTELLT, EINSCHLIEßLICH DER GARANTIE ZUR BENUTZUNG FÜR DEN VORGESEHENEN ODER EINEM BESTIMMTEN ZWECK SOWIE JEGLICHER RECHTSVERLETZUNG, JEDOCH NICHT DARAUF BESCHRÄNKT. IN KEINEM FALL SIND DIE AUTOREN ODER COPYRIGHTINHABER FÜR JEGLICHEN SCHADEN ODER SONSTIGE ANSPRÜCHE HAFTBAR ZU MACHEN, OB INFOLGE DER ERFÜLLUNG EINES VERTRAGES, EINES DELIKTES ODER ANDERS IM ZUSAMMENHANG MIT DER SOFTWARE ODER SONSTIGER VERWENDUNG DER SOFTWARE ENTSTANDEN.