1. Einleitung

In diesem Arbeitsblatt sehen wir uns an, welche Vuetify Komponenten für die Navigation und die Darstellung fixer Inhalte auf der Webseite verwendet werden können. Unter *Navigation* verstehen wir bei Single Page Application (SPA) client-side Routing. Hier mit dem Vue Router. Natürlich kannst du auch mit einem **`router-link`** jederzeit von anderen Komponenten aus Inhalte ändern.

Komponente	Zweck
`v-system-bar`	Kleine Statusinformationen wie zum Beispiel: WLAN Pegel, Anzahl ungelesener E-Mails, etc.
`v-app-bar`	Logo, Titel der Seite, Menü-Links, etc.
`v-main`	Content (oft mit **`router-view`**).
`v-navigation-drawer`	Menü-Links (zum Beispiel bei mobiler Ansicht), und zusätzliche Informationen. Der Drawer kann links (Default) oder rechts angezeigt werden.
`v-footer`	Informationen, auf die der User von jeder Seite innerhalb der App zugreifen möchte. Wird eher selten verwendet.
`v-bottom-navigation`	Alternative zum Drawer, eher für Icon Leiste zur Navigation.

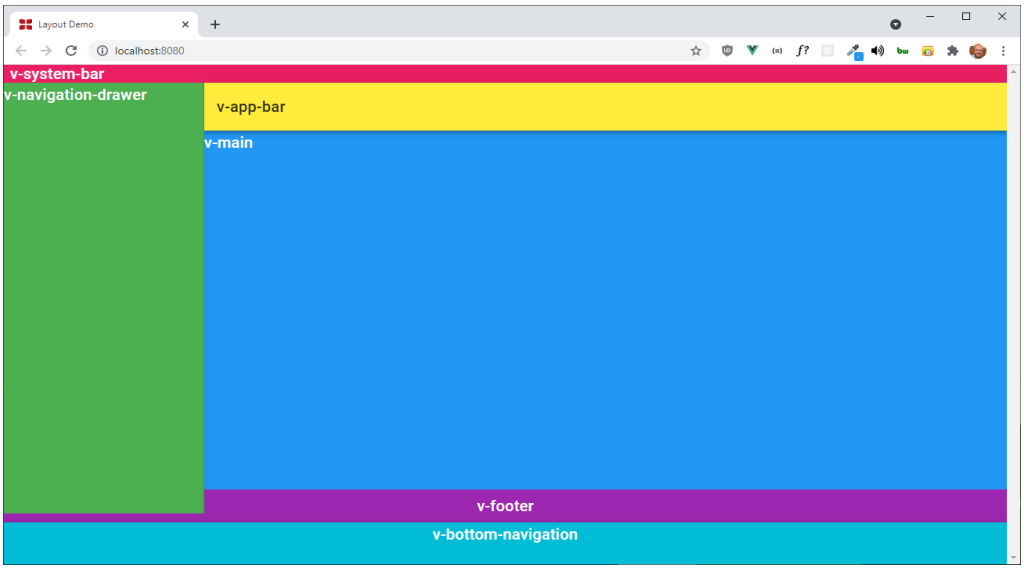
Die Basisstruktur der Komponenten ist wie folgt (mit Vuetify Farben zur besseren Erkennbarkeit):

```
<v-app>
  <v-navigation-drawer class="green" app>
    <span class="white--text title">v-navigation-drawer</span>
  </v-navigation-drawer>

  <v-system-bar class="pink" app>
    <span class="white--text title">v-system-bar</span>
  </v-system-bar>

  <v-app-bar class="yellow" app>
     <span class="title">v-app-bar</span>
  </v-app-bar>

  <v-main class="blue">
    <span class="white--text title">v-main</span>
  </v-main>

  <v-footer class="purple" app>
   <span class="white--text title mx-auto">v-footer</span>
  </v-footer>

  <v-bottom-navigation class="cyan" app>
    <span class="white--text title">v-bottom-navigation</span>
  </v-bottom-navigation>
</v-app>
```

Beachte das Attribut **`app`**, das von dir angegeben werden sollte und die Definition außerhalb von **`v-main`**, aber innerhalb von **`v-app`**!

Ich habe absichtlich keine Margins und kein Paddings verwendet, sondern die Standardposition. In der Darstellung auf der nächsten Seite siehst du zum Beispiel, dass **`v-footer`** und **`v-bottom-navigation`** automatisch Inhalte zentrieren, was

in Hinblick ihres Zweckes Sinn ergibt. Im Gegensatz dazu beginnt der Inhalt von `v-main` links oben. Die Komponente `v-app-bar` wiederum zentriert in der Vertikalen und hat einen kleinen Abstand von Links.



Aufgabe 1: Ist die Reihenfolge, in der die Komponenten im obigen Codebeispiel aufgeführt werden, egal?

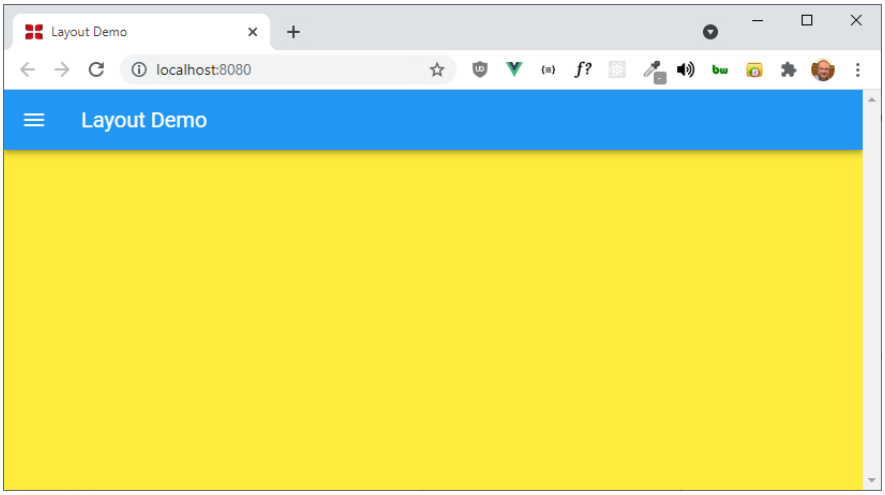
2. Die Komponenten v-app-bar und v-toolbar

Die Komponente `v-app-bar` ist das wichtigste Element zur Navigation. Doch es gibt zwei Arten von Navigation Bars in Vuetify, nämlich `v-app-bar` und `v-toolbar.`

Was ist der Unterschied? `v-app-bar` ist für die Hauptnavigation auf der Webseite gedacht. Daher hat sie zusätzliche Eigenschaften wie Scroll-Animation und mehr Props. `v-toolbar` ist eine Leiste, die überall dort verwendet werden kann, wo man mehrere mögliche Aktionen darstellen möchte (denke an das Werkzeugpanel bei einem Grafikprogramm, oder einer Landkarte (Pins, Search,....).

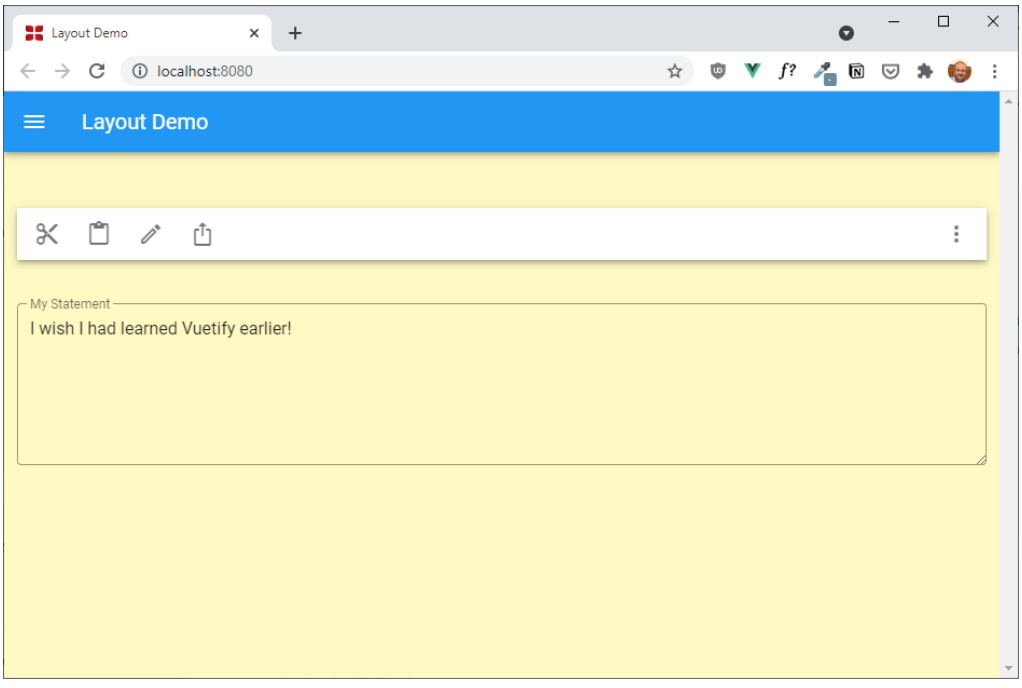
Hier ein Beispiel mit dem beliebten Hamburger Icon (`v-app-bar-nav-icon`) und dem Titel der Webseite.

```
</v-app>
  <v-app-bar class="blue" app>
    <v-app-bar-nav-icon class="white--text"></v-app-bar-nav-icon>
    <v-app-bar-title class="white--text">Layout Demo</v-app-bar-title>
  </v-app-bar>
  <v-main class="yellow"> </v-main>
</v-app>
```



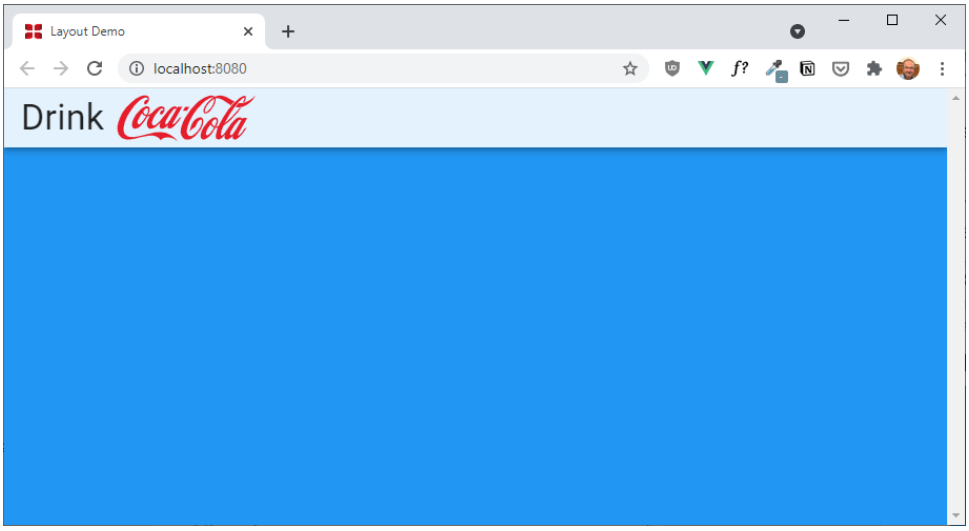
Ein Beispiel: **`v-app-bar`** und **`v-toolbar`**:

```
<v-app>
  <v-app-bar class="blue" app>
    <v-app-bar-nav-icon class="white--text"></v-app-bar-nav-icon>
    <v-app-bar-title class="white--text">Layout Demo</v-app-bar-title>
  </v-app-bar>
  <v-main class="yellow lighten-4">
    <v-container class="mt-10" style="height: 300px">
      <v-toolbar dense>
        <v-btn icon>
          <v-icon>mdi-content-cut</v-icon>
        </v-btn>
        <v-btn icon>
          <v-icon>mdi-content-paste</v-icon>
        </v-btn>
        <v-btn icon>
          <v-icon>mdi-pencil-outline</v-icon>
        </v-btn>
        <v-btn icon>
          <v-icon>mdi-export-variant</v-icon>
        </v-btn>
        <v-spacer></v-spacer>
        <v-btn icon>
          <v-icon>mdi-dots-vertical</v-icon>
        </v-btn>
      </v-toolbar>
      <v-textarea
        outlined
        class="mt-10"
        label="My Statement"
        value="I wish I had learned Vuetify earlier!"
      ></v-textarea>
    </v-container>
  </v-main>
</v-app>
```



Layout Problem 1: Du möchtest ein Logo in der **`v-app-bar`** unterbringen. Das Logo soll in die **`v-app-bar`** passen.

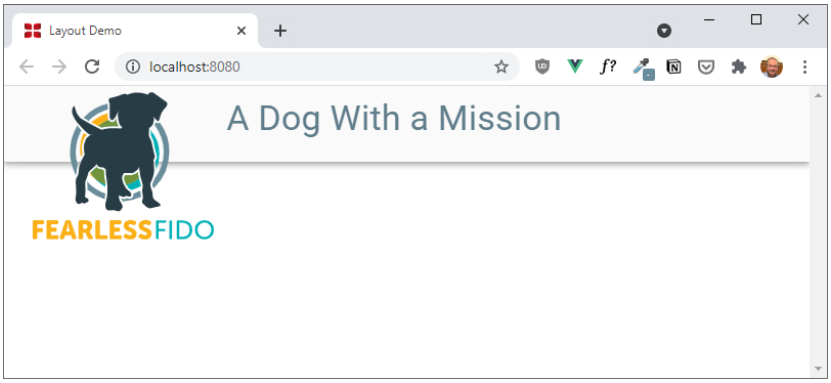
Nimm am besten eine svg Grafik, sonst wirkt das Logo unscharf. Es ist außerdem wichtig, dass du **`v-img`** verwendest und eine maximale Höhe und Breite festlegst, sonst läuft das Bild vertikal über die Navigation und horizontal über den Titel (kann manchmal gewünscht sein, siehe unten!). Verwende zudem **`contain`**, um das Seitenverhältnis zu erhalten.

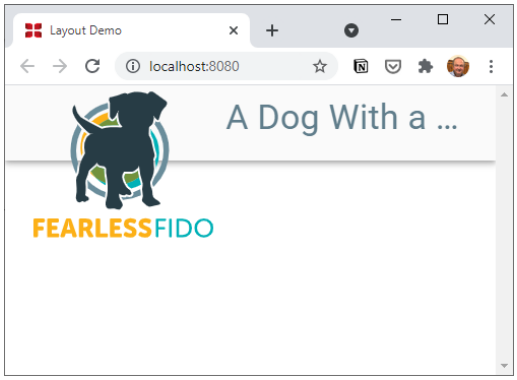



```
<v-app>
  <v-app-bar class="blue lighten-5" app>
    <span class="text-h4">Drink</span>
    <v-img class="mx-2" src="images/logo.svg" max-height="140" max-width="140" contain></v-img>
  </v-app-bar>
  <v-main class="blue"> </v-main>
</v-app>
```

Layout Problem 2: Du möchtest ein Logo in der **`v-app-bar`** unterbringen. Das Logo soll über die **`v-app-bar`** ragen.

Verwende statt **`v-img`** ein normales **`img`** Tag und setze die gewünschte Höhe und Margin.



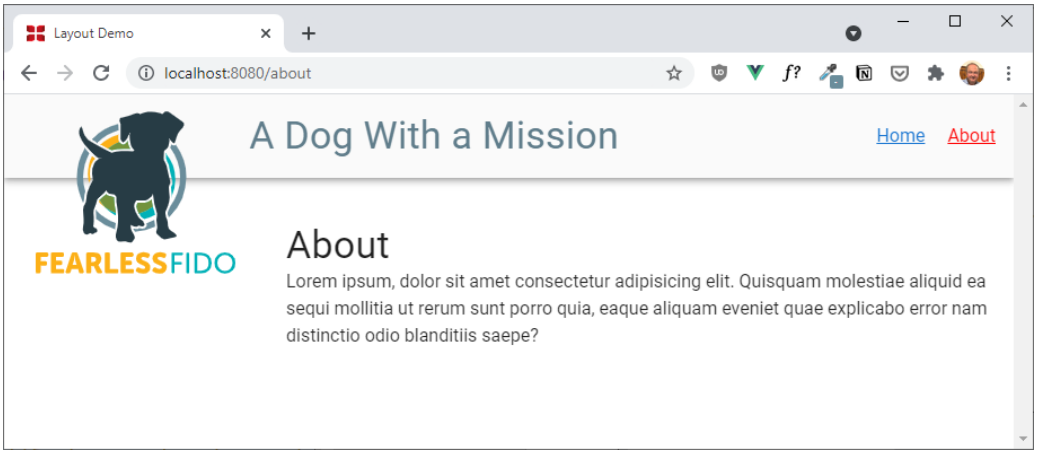


Bonus Tipp: Wenn wenig Platz ist, kannst du einen Text automatisch mit **`text-truncate`** verkürzen.

```
<v-app-bar class="grey lighten-5" height="74" app>
  <img class="mx-2" src="images/fearlessfido-logo.svg" style="height:150px; margin-top:100px" />
  <span
    class="text-truncate text-h4 blue-grey--text darken-1"
    style="font-family: Poppins"
    >A Dog With a Mission</span
  >
</v-app-bar>
<v-main>
</v-main>
```

Um den Inhalt in **router-view** auszutauschen, kannst du bekanntlich **router-link** verwenden. Das Property **active-class** erlaubt dir den aktiven Link zu stylen. Die mit **active-class** spezifizierte Klasse wird dynamisch dem aktiven Link hinzugefügt. Um ein exaktes Matchen der Route zu erreichen, musst du noch das Property **exact** angeben.

```
<v-app>
  <v-app-bar class="grey lighten-5" height="74" app>
    <img class="mx-2" src="images/fearlessfido-logo.svg" style="height:150px; margin-top:100px" />
    <span class="text-truncate text-h4 blue-grey--text darken-1" style="font-family: Poppins"
      >A Dog With a Mission</span
    >
    <v-spacer></v-spacer>
    <router-link active-class="red--text" to="/" exact>Home</router-link>
    <router-link active-class="red--text" to="/about" exact class="ml-5">About</router-link>
  </v-app-bar>
  <v-main>
    <div class="d-flex">
      <div style="width:250px"></div>
      <div class="mt-10" style="flex:1"><router-view></router-view></div>
    </div>
  </v-main>
</v-app>
```



Ich habe herausgefunden, dass Vuetify einen Bug hat und wenn du mit **class=""** eine Farbe definierst, überschreibt sie auch die in **active-class** definierten Werte! In dem Fall musst du zusätzlich deinen Default-Style mit CSS selbst definieren (**router-link** wird zu **a** Tag).

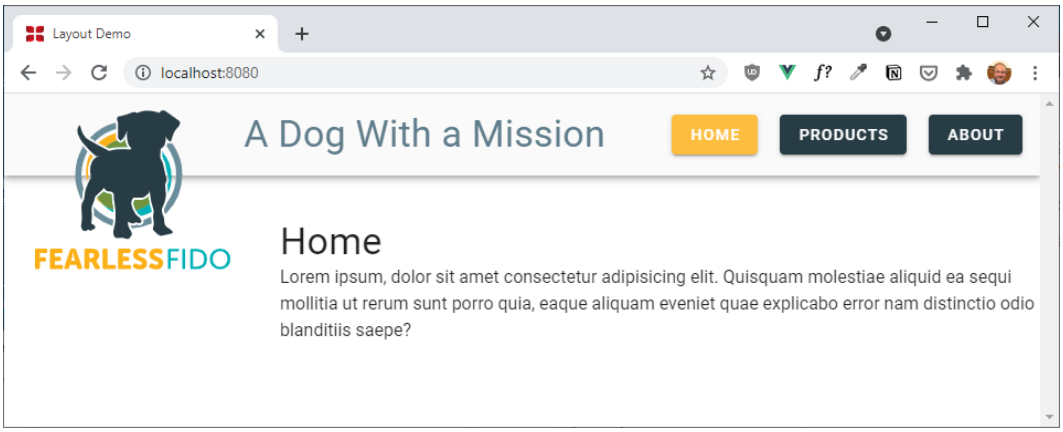
Beispiel:

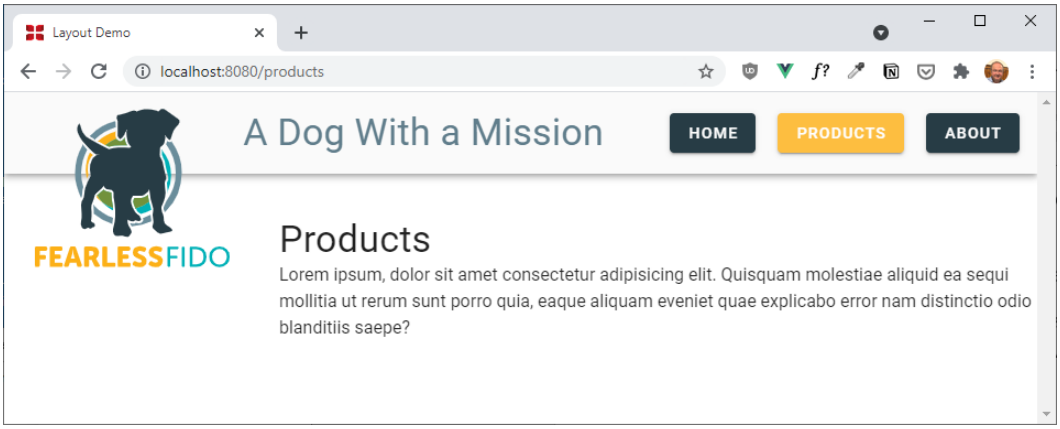
```
<style scoped>
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@500&display=swap');

a:not(active-class) {
  color: grey !important;
}
</style>
```

In dem obigen Beispiel habe ich den linken Bereich mit einem **div** geschützt, das 250px breit ist, sonst schiebt sich der Text in das Logo, wenn wenig Platz ist.

Die folgenden Vuetify Komponenten besitzen alle das Property **to** vom **router-link**: **v-list-tile**, **v-btn**, **v-card.**

Aufgabe 2: Erstelle eine ähnliche Seite wie *Fearless Fido* und verwende statt dem Tag **router-link** das Tag **v-btn**. Style den aktiven Button (Link), sodass der User weiß was ausgewählt wurde!





3. Die Komponente v-navigation-drawer

Manchmal wird auch eine Sidebar verwendet. Die Komponente heißt bei Vuetify **`v-navigation-drawer.`** Die Komponente hat sehr viele Props, am besten liest du mal hier:

https://vuetifyjs.com/en/components/navigation-drawers/

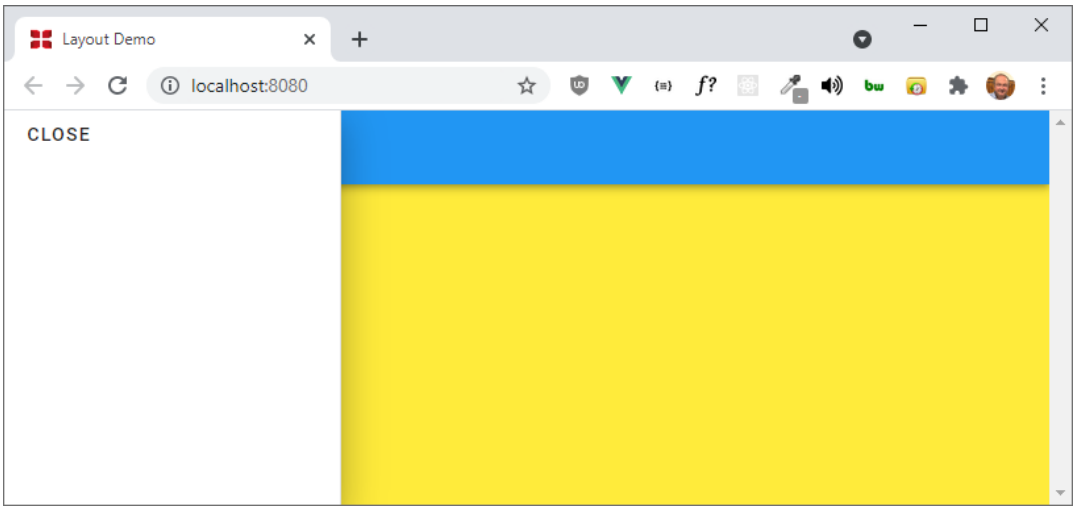
Das untenstehende Beispiel schaltet das Abdunkeln der anderen Komponenten aus (**`hide-overlay`**). Das ist natürlich Geschmacksache (ich finde das Abdunkeln nur gut bei einem Dialog).

Das **`v-model`** Attribut **`drawer`** wird verwendet, um die Sichtbarkeit ein und auszuschalten. Durch Klicken auf das **`v-app-bar-icon`** kann der Drawer ein und ausgeblendet werden.

Durch einen Bug (?) öffnet sich manchmal der Drawer wenn die Fenstergröße sich ändert, daher habe ich den Resize-Watcher ausgeschaltet (**`disable-resize-watcher`**).

Da sich in unserem Fall in der mobile Ansicht der Drawer über die Appbar legt, gibt es im Drawer einen **`Close`** Button. Auch durch Klicken außerhalb des Drawers kann dieser geschlossen werden.

```
<v-app>
  <v-app-bar class="blue" app>
    <v-app-bar-nav-icon @click.stop="drawer = !drawer" class="white--text"></v-app-bar-nav-icon>
    <span class="white--text title">Layout Demo</span>
  </v-app-bar>
  <v-navigation-drawer app hide-overlay disable-resize-watcher v-model="drawer">
    <v-btn text @click="drawer = false">Close</v-btn>
  </v-navigation-drawer>
  <v-main class="yellow">
  </v-main>
</v-app>
```

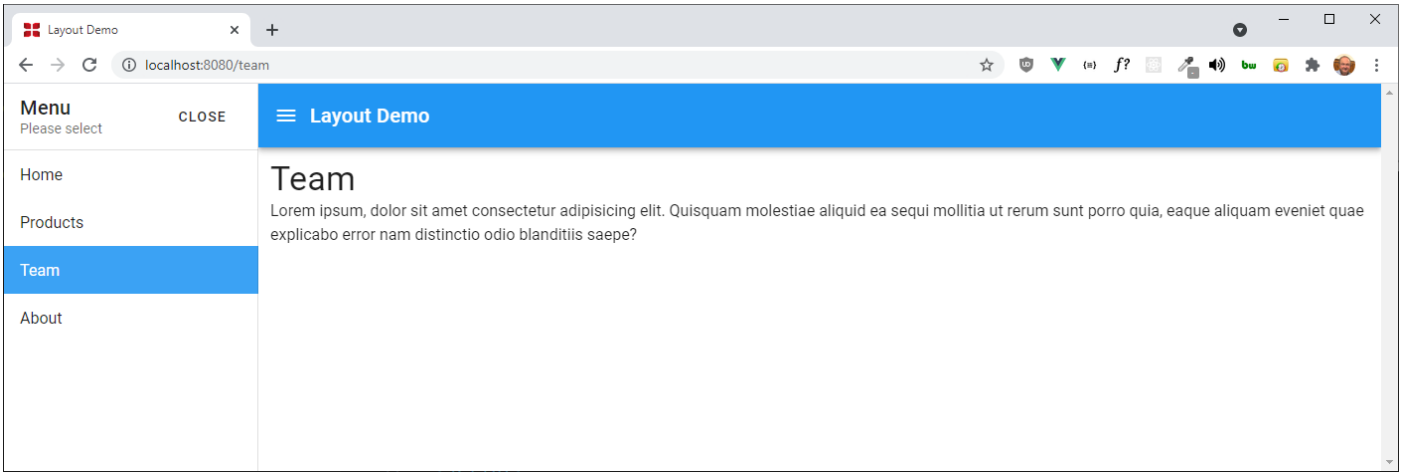


Aufgabe 3: Erstelle vier Routen (`Home`, `Products`, `Team`, `About`) und verlinke im Drawer auf diese Routen. Verwende in den Views Blindtext.

Tips:

- Verwende `v-list-item-group` mit `active-class,` um den gewählten Link hervorzuheben.
- Verwende `v-list-item` und dort verlinke dort mittels `to`
- Für den Header siehe: https://vuetifyjs.com/en/components/navigation-drawers/#usage
- Erstelle dir ein Array mit den Daten für die Routen

```
<v-list-item link v-for="(route, i) in routes" :key="i" :to="route.link">
//  . . .
</v-list-item>
```

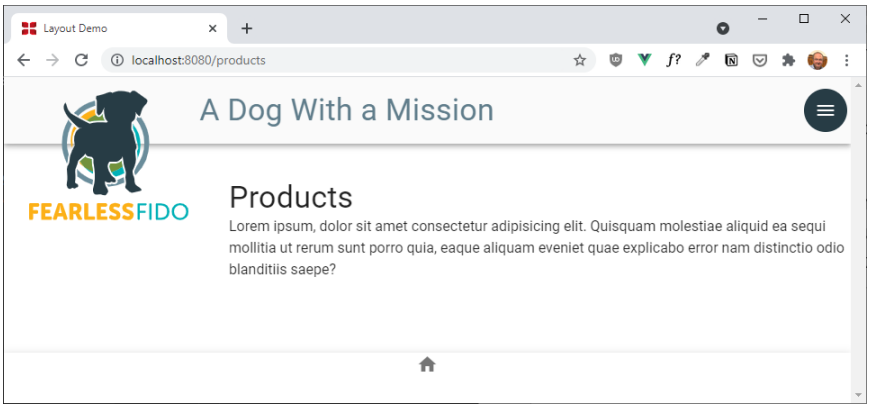


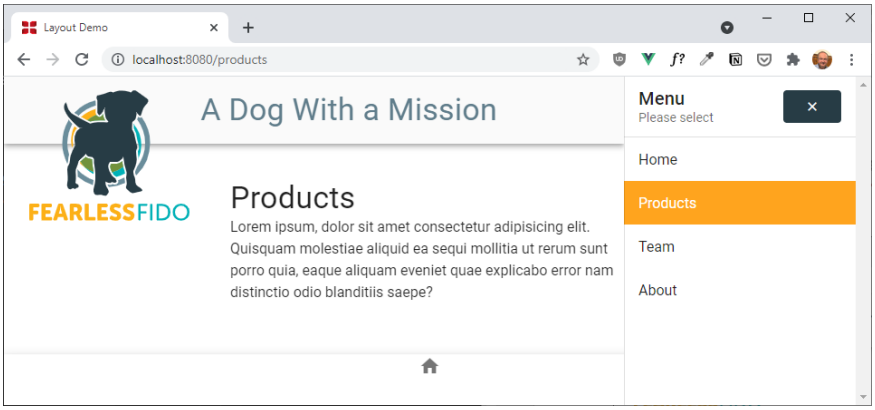
Der Drawer kann auch verwendet werden, um Links aus der v-app-bar anzuzeigen, wenn diese aufgrund der Bildschirmauflösung nicht mehr genug Platz haben. Eine Variante wäre dann das Hamburger Icon erst anzuzeigen, wenn die Links versteckt werden.

Aufgabe 4: Erstelle eine Version deiner Seite von **Aufgabe 2** mit einer Sidebar auf der rechten Seite mit folgender Funktionalität:

Wird die Seite in den Auflösungen **xs**, **sm** und **md** dargestellt, soll statt der Buttons ein Hamburger Icon angezeigt werden. Klickt der Benutzer auf das Icon wird rechts eine Sidebar zur Navigation eingeblendet, das Hamburger Icon verschwindet. Klickt der Benutzer auf das Close Icon in der Sidebar, verschwindet die Sidebar und das Hamburger Icon erscheint. Ab Auflösung lg verschwindet das Hamburger Icon und die Buttons erscheinen.

Wenn du Lust hast, verwende auch `v-bottom-navigation`. Zum Beispiel, um zur Startseite zu springen!





Die Sidebar kann auch in einer Minivariante dargestellt werden. In diesem Fall kann sie mit dem Property `mini-variant` und dem Modifier `sync` abhängig von einer Variablen auf- und zugeklappt werden. Oft in Kombination mit dem Property `permanent`.

Pro Aufgabe 5: Für den Beispiel aus Aufgabe 2, erstelle eine Variante, wie auf der nächsten Seite gezeigt.

Ein Klick auf die Sidebar erweitert diese, sodass die weitere Beschreibung sichtbar wird.

Hinweis: In diesem Fall ist der `v-navigation-drawer` nicht Teil des Appframes, sondern Teil von `v-main` (daher kein `app` Property)!

Weiter Varianten: expand on hover (finde ich ein bisschen nervig, aber ist Geschmacksache), Positionierung an einer anderen Stelle auf dem Bildschirm.

Robert Baumgartner