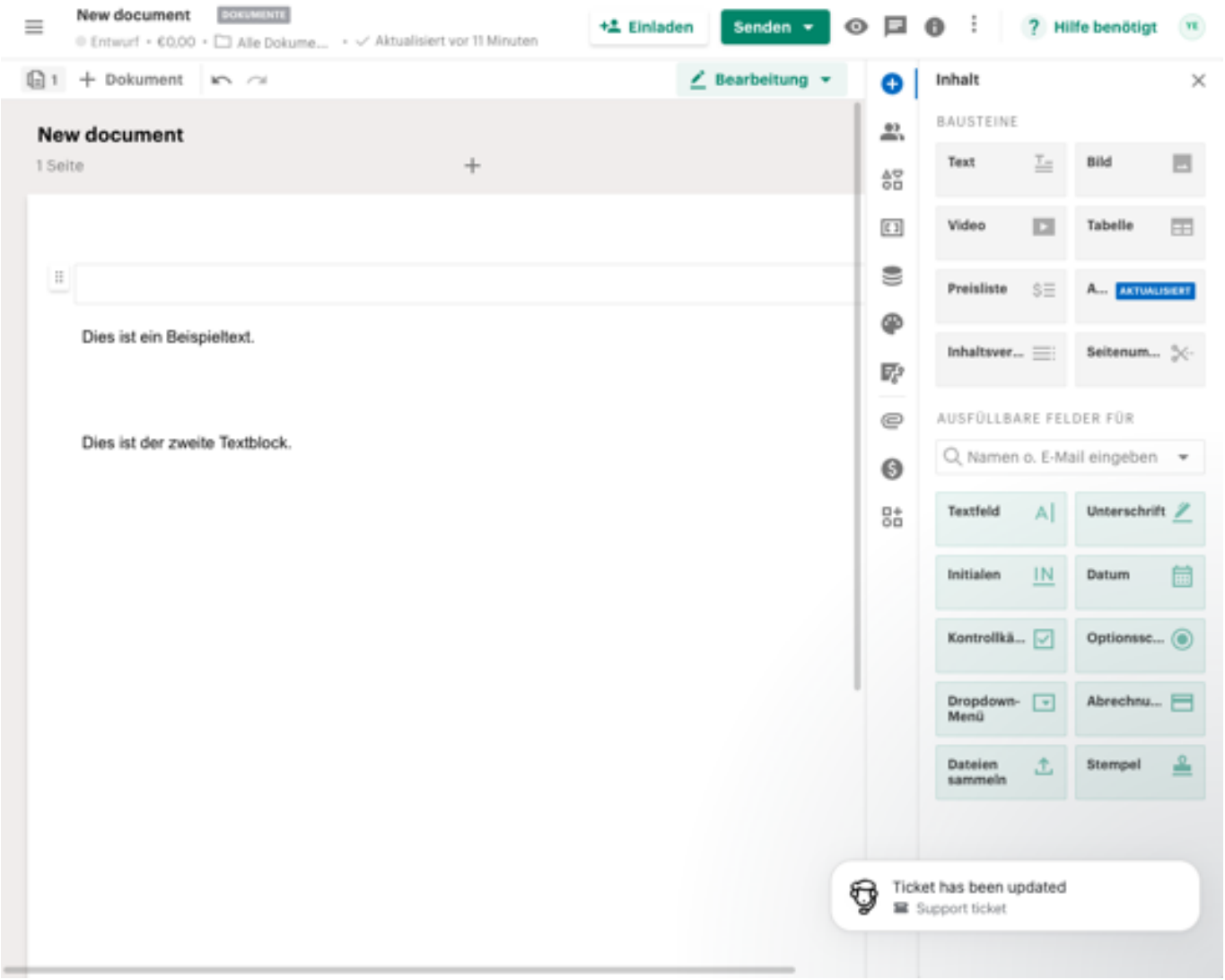
Reverse Engineering: PandaDoc Dokument-Editor

Dieses Dokument fasst alle Beobachtungen zur PandaDoc-Oberflaeche zusammen, inklusive Dashboard, Wizard, Editor, Text-Bloecken, Plus-Icon/Quick-Add-Menue und Styling (Abstaende, Groessen, Hover-Effekte). Ziel: Du sollst damit den Editor fuer dein eigenes Produkt nachbauen koennen (HTML/CSS/JS-Struktur).

Abbildung 1: Editor-Uebersicht



1. Gesamtaufbau der Anwendung

Die Web-App laeuft als Single Page Application (SPA) mit drei grossen Bereichen:
1) Dashboard (Liste aller Dokumente)
2) Erstellungs-Wizard (Schritte von Vorlage bis Empfaenger)
3) Dokument-Editor (zentraler Canvas mit Seitenleiste rechts).

Der Editor selbst ist blockbasiert: Jeder Inhalt (Text, Bild, Tabelle usw.) ist ein Block. Blocks sind im DOM in einer Tree-Struktur organisiert und werden ueber ein contenteditable-basiertes Rich-Text-Framework (z.B. Slate.js) gerendert.

2. Dokument-Dashboard (Listenansicht)

Header:

- Linke Seite: Suchfeld mit Placeholder "Dokumente durchsuchen".
- Rechts davon: Filter/Zeitleiste, Benutzer-/Kontomenue, gruener Button "Dokument" (mit Dropdown fuer Dokument, Formular usw.).

Tabs unter dem Header:
- "Neueste", "Alle Dokumente", "Von mir erstellt", "Archiviert" und ggf. weitere im Dropdown.
- Aktiver Tab: gruene Unterstreichung (ca. 2 px), Tab-Text in dunkler Schrift.

Dokumentenliste:
- Tabelle mit Spalten: Name, Status, Betrag, Geaendert.
- Jede Zeile ist als klickbarer Link aufgebaut (`<a data-testid="lister-entity-row">`).
- In der Zeile:
  * Titel (z.B. "New document") als Blockelement.
  * Untertitel (z.B. "Keine Empfaenger").
  * Avatar-Kreis mit Initialen (data-testid="avatar").
  * Datum/Zeit als `<p>`.
  * Rechts ein Icon-Button mit data-testid="table-actions-handle` fuer Kontextmenue.

3. Dokumenterstellung per Wizard

Der Wizard oeffnet sich als Modal ueber dem Dashboard.

Struktur:
- Linke Spalte: Navigation "Schnellzugriff", "Meine Vorlagen", "Mit mir geteilt" plus Importoptionen (Datei-Upload, Google Drive, Dropbox, OneDrive).
- Oberhalb der Mitte: Step-Indikator mit den Schritten:
  1) "Los geht's" (Vorlagenwahl)
  2) "Empfaenger hinzufuegen"
  3) "Inhalt ueberpruefen".
- Hauptbereich: Kacheln mit Vorlagen. Eine Kachel ist "Blank document" fuer ein leeres Dokument.

DOM-Hinweise:
- Vorlagenkacheln sind `<li>`-Elemente mit Titel und ggf. Badges.
- Buttons fuer Navigation verwenden data-testid, z.B. `add_recipients_step__continue_button`.
- "Blank document" ist eine spezielle Kachel mit eigenem data-Attribut, die direkt den Editor mit einer leeren Seite oeffnet.

4. Editor: Layout und Panels

Der Editor teilt sich in drei Bereiche:

A) Kopfzeile (Top-Bar):
- Links: Dokumenttitel (z.B. "New document") als inline-editierbares Feld (vermutlich ein `<div contenteditable>` oder ein spezieller Titel-Input).
- Daneben: Status-Label ("Entwurf"), Gesamtbetrag (z.B. "EUR 0,00") und Info "Aktualisiert vor X Minuten".
- Rechts: Buttons "Einladen", "Senden" (gruene Primary-Buttons) sowie ein Icon-Menue fuer Dokumentinformationen, Aktivitaetslog, Kommentare usw.

B) Mittlerer Bereich (Seiten-Canvas):

- Links im Canvas: Mini-Leiste fuer Seiten (z.B. "1 Seite" und ein Plus zum Hinzufuegen neuer Seiten).
- Hauptseite: Weiss hinterlegte Seite, zentriert mit Schatten/Border, reagiert auf Scroll.
- Innerhalb der Seite liegen Block-Container (Text, Bild etc.).

C) Rechte Seitenleiste (Panel "Inhalt"):
- Tabs am oberen Rand (Icons): Inhalt, Empfaenger, Content-Bibliothek, Variablen, Daten, Design, Workflow.
- Tab "Inhalt" zeigt zwei Gruppen:
  * "Bausteine" (Text, Bild, Video, Tabelle, Preisliste, Angebotsersteller, Inhaltsverzeichnis, Seitenumbruch).
  * "Ausfuellbare Felder" (Textfeld, Unterschrift, Initialen, Datum, Kontrollkaestchen, Optionsschaltflaechen, Dropdown-Menue, Abrechnungsinformationen, Dateien sammeln, Stempel).

5. Content-Bloecke (Textblock-Implementierung)

Jeder Content-Block wird durch einen Frame mit Drag-Handle dargestellt.

DOM-Struktur eines Textblocks (vereinfacht):
- Aeusserer Container: `<div class="pd-block" data-testid="content-base" data-block-id="...">`
- Links im Container: Drag-Handle (zwei vertikale Punkte) als eigenes `<div>`.
- Inhalt: `<div contenteditable="true" role="textbox" data-slate-editor="true" data-key="16" data-gramm="false">Dies ist ein Beispieltext.</div>`
- Weitere Bloecke haben andere `data-key`-Werte (z.B. `data-key="28"` fuer den zweiten Block).

Verhalten:
- Klick in den Block setzt den Cursor, die Editor-Toolbar fuer Text (Schriftart, Groesse, Bold, Align usw.) wird eingeblendet.
- Hover ueber den Block zeigt oben eine Floating-Toolbar mit Block-Aktionen: Duplizieren, Kopieren, Ausschneiden, Kommentar, Eigenschaften, Sperren, Loeschen. Buttons tragen data-testid wie `content-base-copy`, `content-base-cut`, `content-base-delete`.
- Der Block kann ueber Drag-Handle per Drag&Drop verschoben werden (JavaScript haengt sich an die data-block-id).

6. Plus-Icon & Quick-Add-Funktion

Das Plus-Icon ist der zentrale Entry-Point, um neue Bloecke kontextsensitiv zwischen existierenden Bloecken einzufuegen.

Interaktions-Flow:
1) Maus zwischen zwei Bloecken bewegen (oder an den oberen/unteren Rand eines Blocks).
2) Eine duenne horizontale Linie erscheint ueber die gesamte Breite des Inhaltsbereichs.
3) In der Mitte der Linie erscheint ein kleines blaues Plus-Symbol mit rundem Hintergrund.
4) Klick auf das Plus oeffnet ein Quick-Add-Menue.
5) Auswahl einer Option (z.B. "Text") erzeugt einen neuen Block an genau dieser Stelle.

Quick-Add-Menue:
- Abschnitt "Bibliotheken": Content-Bibliothek, Image-Bibliothek, Canva.
- Abschnitt "Schnell hinzufuegen": Text, Bild, Video, Tabelle, Preisliste, Angebotsersteller, Inhaltsverzeichnis, Seitenumbruch.
- Jede Option hat eine Nummer (1-8) als Shortcut-Hinweis.
- Im DOM vermutlich `<div class="quick-add-menu">` mit Eintraegen, die data-testid wie

`quick-add-content__menu-item--text` besitzen.

DOM-Aenderung beim Einfuegen:
- Beim Klick auf "Text" wird zwischen den beiden bestehenden Block-Containern ein neuer Block-Container mit einem frischen `data-key` in das Slate-Tree eingefuegt.
- Der neue Block startet mit leerem Paragraphen; der Cursor wird fokussiert.

7. Styling: Abstaende, Groessen, Hover-Effekte

Allgemeines Layout:
- Seitenleiste rechts: feste Breite ca. 220 px, vertikale Trennlinie zum Canvas.
- Canvas: zentrierte Seite mit max. Breite einer DIN-A4-Seite, leichte Schattenkante.

Block-Layout:
- Innenabstand im Block (padding): ca. 6-8 px oben/unten, etwas mehr links (wegen Drag-Handle).
- Drag-Handle-Breite: ca. 8 px, mit zwei gestrichelten Punkten oder Linien.
- Vertikaler Abstand (margin) zwischen Bloecken: ca. 12-16 px.
- Beim Hover: Block-Rand wird leicht grau hervorgehoben, die horizontale Einfuege-Linie taucht auf.

Quick-Add-Menue:
- Breite ca. 180 px, weisser Hintergrund, leichter Box-Shadow.
- Eintraege: ca. 30 px hoch, 4-5 px Abstand vertikal, Nummer rechts in grauer Schrift.

Seitenleiste "Bausteine" und "Ausfuellbare Felder":
- Raster mit 2 Spalten.
- Bausteine: ca. 85x45 px pro Kachel, Icon oben, Text unten, leichter Hover-Effekt.
- Ausfuellbare Felder: Kacheln ca. 100x40 px, oft mit farbigem (hellgruenem) Rand.

Dashboard:
- Tab-Leiste: Tabs mit ca. 20 px Abstand, aktive Unterstreichung (2 px, gruen).
- Tabellenzeilen: Hoehe ca. 56 px, hauchduenne horizontale Trennlinie.

8. Architektur-Vorschlag fuer dein eigenes Produkt

Um ein aehnliches Produkt zu bauen, kannst du folgende Struktur nutzen (vereinfacht):

HTML-Skelett (Konzept):
- `<div class="app-shell">`
  * `<header class="topbar">` (Dokumenttitel, Status, Buttons)
  * `<main class="editor-layout">`
    - `<aside class="page-strip">` (Seitenuersicht)
    - `<section class="page-canvas">`
      + Liste von Block-Containern
      + Zwischen den Bloecken unsichtbare Einfuege-Linien, die beim Hover sichtbar werden
    - `<aside class="sidebar">` (Bausteine, Felder)

Block-Komponente (z.B. React):
- Props: id, type, data, index.
- Render: `div.block` mit Handle, contenteditable-Wrapper, Floating-Toolbar.

- Events: onMouseEnter -> zeigt Plus-Linie; onClickPlus -> oeffnet Quick-Add-Menue; onSelectMenuItem -> fuegt neuen Block in Blockliste ein.

State-Management:
- Zentrale Liste von Block-Objekten (z.B. [{id, type, data}, ...]).
- Quick-Add Menue bekommt Info, an welcher Index-Position eingefuegt werden soll.
- Bei Insert: `blocks.splice(insertIndex, 0, newBlock)`; Editor rendert neu.

Rich-Text:
- Verwende ein Framework wie Slate.js oder ProseMirror, um contenteditable sauber zu handhaben.
- Speichere fuer jeden Block eine JSON-Repraesentation (z.B. Slate-Node-Tree), nicht nur plain HTML.

CSS-Prinzipien:
- Nutze CSS-Variablen fuer Abstaende und Groessen (z.B. `--block-padding-y: 8px;`).
- Definiere Utility-Klassen fuer Hover-Status (`.block--hover`, `.insert-line--active`).
- Achte auf konsistenten zentrierten Canvas und feste Sidebar-Breite.