# Tapetendesigner – Dokumentation

Stand: 12.11.2025
Projekt: Shopify Wallpaper Designer

# Teil 1 – Einfache Erklärung (für Anwender)

Der Tapetendesigner erlaubt dir, eine individuelle Fototapete anhand deiner Wandmasse zu gestalten. Du gibst zuerst die Wandbreite und Wandhöhe ein. Das System berechnet automatisch das Druckmass (ganze Bahnen) und zeigt dir die sichtbare Bildbreite (Wand + 10 cm Sichtreserve).
Die schraffierte graue Fläche zeigt das Übermass: Dieser Bereich wird nicht bedruckt, weil wir nur ganze Bahnen verwenden.
Du kannst das Bild hochladen (JPG, PNG, TIFF, PDF, SVG, EPS). Danach kannst du mit der Maus das Motiv verschieben und mit dem Zoomregler vergrößern/verkleinern.
Die gelben gestrichelten Linien zeigen die Bahnen-Grenzen. So siehst du, wo später geschnitten/gesetzt wird.
Mit dem Button "Übermass links/rechts" entscheidest du, auf welcher Seite das unbedruckte Übermass sein soll.
Der rote Rahmen zeigt das Wandmass. Innerhalb dieses Rahmens muss dein Motiv Platz finden. Das Bild selbst wird nur im sichtbaren Bereich (Wand + 10 cm) dargestellt.
Wenn alles passt, generierst du einen Code (Konfigurationscode). Mit diesem Code kannst du später im Shop oder bei Nachfragen genau dieselbe Konfiguration wieder laden.
Der Button für den PDF-Proof erstellt ein "Gut zum Druck" Dokument: Das PDF zeigt Ausschnitt, Maße, Bahnenlinien und Übermass so wie im Designer. Dieses Dokument dient als finale Freigabe.
Im PDF ist der schraffierte Bereich immer weiß hinterlegt und nicht bedruckt; so wird sichergestellt, dass kein Teil des Motivs dort versehentlich erscheint.

• Wandmass: Deine eigentliche Wandfläche.
• Bildmass: Wandbreite + 10 cm Reserve – nur dieser Bereich zeigt das Motiv.
• Druckmass: Ganze Bahnenbreite, kann breiter sein als das Bildmass.
• Übermass: Differenz zwischen Druckmass und Bildmass, schraffiert und unbedruckt.
• Zoom: Vergrößert das Motiv – achte auf ausreichende Bildqualität.
• Offsets: Verschieben des Ausschnitts (durch Drag).
• Konfigurationscode: Referenz zum Wiederladen und für die Produktion.

# Teil 2 – Technische Dokumentation (für Entwickler)

Ziel: Verständnis der internen Datenflüsse, Berechnungen und Erweiterungsmöglichkeiten.

# Architektur-Überblick

Frontend (React, Datei: frontend/src/index.js) – enthält die Hauptkomponente FrameDesigner. Hier werden Dimensionen berechnet, Bild geladen, Zoom/Drag/Flip und sichtbare Bereiche gehandhabt.
Backend (Express, pdfProof.js) – erzeugt den PDF-Proof, lädt Bilddaten, berechnet sichtbare Bereiche und zeichnet Clip + Weiß + Schraffur.
Persistenz: Konfigurationen werden als Dateien gespeichert (configStore). Transform-Informationen (zoom, offsetXPct, offsetYPct, naturalWidth/Height) werden mit abgelegt.
Keine serverseitigen Crops mehr: PDF spiegelt 1:1 die UI-Cover-Logik (scale + zoom + offsets + clip).

# Maß-Definitionen

• wall.widthCm (Wandmass) & wall.heightCm
• print.widthCm / print.heightCm (Druckmass = volle Bahnen)
• Bildmass = wall.widthCm + 10 cm (sichtbarer Motivbereich)
• extraWhiteWidthCm = print.widthCm "  (wall.widthCm + 10)
• overageSide: left | right – bestimmt Position des Übermaßbereichs

# Transform-Daten

zoom: Multiplikator auf die Cover-Skalierung (Basis: sichtbares Bildmass vs. Wandhöhe).
offsetXPct / offsetYPct: Relative Mittelpunkt-Offsets (0..1) bezogen auf die gezoomte, gescalte Naturalgröße.
flipH / flipV: Horizontale / vertikale Spiegelung – im PDF per sharp.flop()/flip().
naturalWidth / naturalHeight: Originalbildgröße (Pixel), für Qualitäts- und Zoomberechnung wichtig.

# Positionsberechnung (UI & PDF)

1. Berechne scale s = max(visibleWidthPx / naturalWidthPx, frameHeightPx / naturalHeightPx); deckel s auf 1.
2. Effektive Zeichengröße: drawW = naturalWidthPx * s * zoom; drawH analog.
3. Mittelpunkt im sichtbaren Bereich: centerX = offsetXPct * drawW; centerY analog.
4. Linke obere Bild-Ecke: posX = clipLeft + clipWidth/2 "  centerX; posY = frameTop + clipHeight/2 " centerY.
5. Clip-Rechteck deckt Bildmass ab; außerhalb liegt Übermass (weiß + Schraffur).

# PDF-spezifische Schritte (pdfProof.js)

Lädt Bildpuffer (preview oder originalUrl).
Wendet Flip an (sharp.rotate/flop/flip).
Berechnet extraWhiteCm Fallback: (printW - (wallW + 10)).
Berechnet Clip-Koordinaten abhängig von overageSide.
Weißes Rechteck als Unterlage im Clip, dann Bild mit drawW/drawH an posX/posY.
Übermassbereich: doppelte Weißfüllung + diagonale Schraffur + dünne Edge-Linie.
Bahnenlinien: gestrichelte vertikale Linien relativ zum sichtbaren Bereich (stripWidthCm * i).

# Erweiterungshinweise

• Zusätzliche Effekte (z.B. Farbfilter) im Frontend über eine Canvas-Layer vor dem Upload anwenden.
• Weitere Ausgabeformate: eigenen Endpoint erstellen, pdfProof-Logik wiederverwenden, ggf. Rasterisierung beibehalten.
• Qualitätshinweis verfeinern: effW/effH vs. Mindest-Pixel pro cm dynamisch loggen.
• Testing: Unit-Tests für Transform-Berechnung (Offsets bei Flip, Zoom-Grenzen).
• Internationalisierung: Texte in separater JSON und per Key injizieren.

# Fehlerquellen & Debugging

• Versatz nach Reload: Prüfe Rehydration der Offsets (offsetXPct/offsetYPct) + zoom.
• Bild erscheint im Übermass: Clip-Breite korrekt? extraWhiteCm korrekt (print - (wall + 10))?
• Falsche Bahnenlinien: stripsCount oder bahnenbreiteCm fehlen / Rundung in px überprüfen.
• Qualität zu gering: Ursprungsbild prüfen (naturalWidth/Height) und zoom > 1 kritisch hinterfragen.

# Dateien & Einstiegspunkte

frontend/src/index.js  -> FrameDesigner + UI-Logik
backend/services/pdfProof.js  -> PDF-Erzeugung (Proof)
backend/services/configStore.js -> Speichern/Laden von Konfigurationen
backend/index.js  -> API-Routen (Konfig, PDF)
scripts/backup.ps1 / restore.ps1 -> Snapshots

# API Felder (vereinfachter Auszug)

```
POST /config {
  wall: { widthCm, heightCm },
  print: { widthCm, heightCm },
  calc: { mode, bahnenbreiteCm, extraWhiteWidthCm, overageSide, strips },
  transform: { zoom, offsetXPct, offsetYPct, flipH, flipV, naturalWidth, naturalHeight },
  image: { url, preview, originalUrl, filename }
```

}

# Wichtige Konstanten

Bildmass = Wandbreite + 10 cm
extraWhite = Druckbreite - Bildmass
Offsets beziehen sich IMMER auf die gezoomte, gescalte Naturalgröße
Zoom deckelbar (Frontend: 1..3)

# Qualitätshinweis (Berechnung)

effektivePixelBreite = naturalWidth / zoom
erforderlichMin = wandBreiteCm * 15 (z.B. Schwelle für Orange)
Schwellen anpassbar in Disclaimer-Logik pdfProof.js

# Restore dieses Standes

powershell -NoProfile -ExecutionPolicy Bypass -File scripts/restore.ps1 -ZipPath snapshots/20251112-...-pdf-visible-cover-20251112.zip

# Lizenz / Rechte

Alle Bildrechte müssen beim Besteller liegen. Der Code im PDF repräsentiert die Freigabe (Gut zum Druck).