# BlackDress · Operations-Handbuch

**Version**: 1.2 **Erstellt**: 2026-05-08 **Letzte Aktualisierung**: 2026-05-08 abend (Build-Tag A-Z + Email-Worker-Recipe + Test-Suite) **Zielgruppe**: Don alleine, ohne Genesis-Verfügbarkeit **Druckbar**: ja (Markdown → PDF via Pandoc oder Browser-Print)

> ***Goldene Regel**: Wenn du unsicher bist — **STOPP** und warte bis Genesis verfügbar. Diese Anleitung ist für **Routine-Operations**, nicht für riskante One-Off-Aktionen.*

## INHALTSVERZEICHNIS

| | | |
|---|---|---|
| 19 | E2E Test-Suite lokal laufen lassen | 5 Min |
| 20 | Pages-Index-Schnellzugang (für QA-Klick-Test) | 1 Min |

---

## 1 · BESTELLUNG EINSEHEN + STATUS ÄNDERN

**Voraussetzung**: D1 ist deployed (TODO `blackdress-d1-deploy`). Vor Live-Gang.

### Variante A — Cloudflare-Dashboard direkt

1. https://dash.cloudflare.com → Login
2. Workers & Pages → D1 → `blackdress-prod`
3. Tab "Console" → SQL-Query:

```
SELECT * FROM orders ORDER BY created_at DESC LIMIT 20;
```

4. Status ändern:

```
UPDATE orders SET status='shipped', shipped_at=datetime('now'),
tracking_number='ABC123' WHERE order_number='BD-2026-XXXXX';
```

### Variante B — Order-Status-API (wenn gebaut)

- (Phase F-Step E baut interne Admin-Page für Don)
- Aktuell noch nicht verfügbar, Variante A nutzen

### Status-Werte erlaubt

`pending` · `paid` · `processing` · `shipped` · `delivered` · `cancelled` · `refunded`

---

## 2 · VERSAND MARKIEREN

1. Pakete vom Schweizer Post / DHL aufgeben
2. Tracking-Nummer notieren
3. In D1 (siehe Schritt 1):

```
UPDATE orders
SET status='shipped',
    shipped_at=datetime('now'),
    tracking_number='<post-tracking>',
    carrier='Schweizer Post',
    updated_at=datetime('now')
WHERE order_number='BD-2026-XXXXX';
```

4. **Email-Trigger**: passiert automatisch über Worker-Webhook (Phase F)
5. **Falls Worker nicht gebaut**: Email manuell aus `email_templates/shipping_notification.de.html` zusammen-copy-paste-en + via Google-Workspace-Mail an Kunde senden

## 3 · REFUND AUSLÖSEN

**Wichtig**: Refunds gehen über Payrexx-Dashboard (NICHT direkt an Bank), damit der Auditspur erhalten bleibt.

1. https://payrexx.com → Login (Speicherort siehe Zugangs-Inventar §D)
2. Transactions → finde Order via `payment_reference` aus D1:

```
SELECT order_number, payment_reference, total_cents, status
FROM orders WHERE order_number='BD-2026-XXXXX';
```

3. In Payrexx: Transaction → Refund-Button → Betrag (full or partial)
4. Bestätigen → Payrexx initiiert Refund (3-7 Werktage)
5. **In D1** Status updaten:

```
UPDATE orders SET status='refunded', refunded_at=datetime('now') WHERE
order_number='BD-2026-XXXXX';
```

6. **Inventory restock**:

```
-- Pro order_item den variant-stock zurückaddieren
UPDATE inventory SET qty_on_hand = qty_on_hand + (
  SELECT quantity FROM order_items WHERE order_id=(SELECT id FROM orders WHERE
order_number='BD-2026-XXXXX') AND sku=inventory.sku
) WHERE sku IN (SELECT sku FROM order_items WHERE order_id=(SELECT id FROM orders
WHERE order_number='BD-2026-XXXXX'));
```

7. **Cancellation-Email** an Kunden (per `email_templates/cancellation.de.html`)

## 4 · PRODUKT DEAKTIVIEREN / OUT-OF-STOCK

### Variante A — Product komplett verstecken (z.B. discontinued)

1. SSH zum VPS:

```
ssh root@<VPS-IP>
```

2. products.json editieren:

```
cd /root/blackdress
cp data/products.json backups/products_pre_disable_$(date +%Y-%m-%d-%H%M).json
```

3. JSON öffnen, Produkt-Eintrag löschen ODER neuen Flag `"discontinued": true` setzen
4. Site re-deployen:

```
bash /root/deploy_cloudflare.sh
```

### Variante B — Out-of-Stock setzen (Vendor-Lieferung wartet)

- Variant-`qty` auf 0 setzen in products.json (gleicher Pfad wie Variante A)
- UI zeigt automatisch "Aktuell nicht verfügbar"
- ODER (sobald D1 live): `UPDATE inventory SET qty_on_hand=0 WHERE sku='XXX'`

### Variante C — gesamte Produktkategorie unsichtbar (z.B. Saison-Ende)

- In products.json: alle Produkte einer collection-Tag entfernen ODER discontinued setzen
- Re-Deploy
- Cache-Busting: Cloudflare-Pages-Purge im Dashboard

---

## 5 · DSGVO-AUSKUNFTS-ANFRAGE (Art. 15)

**Rechtsanspruch**: jeder Kunde kann Auskunft über alle gespeicherten Personendaten verlangen, kostenfrei innerhalb 30 Tagen.

### Ablauf

1. Kunde mailt an info@blackdress.ch mit Subject "DSGVO-Auskunft"
2. **Identifikation prüfen**: Email-Match mit account ODER ID-Foto-Verifikation
3. SSH zum VPS:

```
cd /root/blackdress
```

4. Customer-Hash berechnen:

```
node -e "
const { hashCustomer } = require('./lib/aes.js');
const fs = require('fs');
const salt = fs.readFileSync('/root/.blackdress_credentials','utf8').match(/
CUSTOMER_HASH_SALT=(.*)/)[1].trim();
hashCustomer(salt, 'kunde@email.ch').then(h => console.log(h));
"
```

5. In D1 alle Einträge sammeln (Hash kopieren aus 4):

```
SELECT * FROM customers WHERE customer_hash='<hash>';
SELECT * FROM orders WHERE customer_id=(SELECT id FROM customers WHERE
customer_hash='<hash>');
SELECT * FROM order_items WHERE order_id IN (SELECT id FROM orders WHERE
customer_id=(SELECT id FROM customers WHERE customer_hash='<hash>'));
SELECT * FROM email_log WHERE customer_id=(SELECT id FROM customers WHERE
customer_hash='<hash>');
SELECT * FROM customer_tags WHERE customer_id=(SELECT id FROM customers WHERE
customer_hash='<hash>');
```

6. **Verschlüsselte Felder dechiffrieren** (Worker oder lokales Node-Script via `lib/aes.js`):

```
node -e "
const { decrypt, decryptJson } = require('./lib/aes.js');
```

```
const aesKey = process.env.BLACKDRESS_AES_KEY;
decrypt(aesKey, '<email_encrypted-blob>').then(console.log);
"
```

7. Alle Felder in **PDF zusammenfassen** (Vorlage `dsgvo_auskunft_template.md` — TODO Phase F)
8. Per Email an Kunde senden inkl. Erklärung was-bedeutet-was

**Frist**: 30 Tage. Bei Komplexität ist Verlängerung um 2 Monate möglich (Art. 12 Abs. 3).

---

## 6 · DSGVO-LÖSCHUNGS-ANTRAG (Art. 17)

**Rechtsanspruch**: Kunde kann Löschung seiner Personendaten verlangen. **Aber**: Order-Daten haben **10 Jahre Aufbewahrungspflicht** (CH OR / EU GoBD). Diese werden anonymisiert, nicht gelöscht.

### Ablauf

1. Kunde mailt info@blackdress.ch "DSGVO-Löschung"
2. Identifikation prüfen (siehe §5)
3. Customer-Hash berechnen (siehe §5)
4. **Hard-Delete der PII** in D1:

```
-- Email + Name + Address-Encrypted nullifizieren
UPDATE customers
SET email_encrypted='[GELÖSCHT_ART17]',
    name_encrypted=NULL,
    deleted_at=datetime('now')
WHERE customer_hash='<hash>';

-- Adressen in orders nullifizieren (Order-Daten selbst bleiben für Steuer)
UPDATE orders
SET shipping_address_encrypted='[GELÖSCHT_ART17]',
    billing_address_encrypted=NULL,
    notes_encrypted=NULL
WHERE customer_id=(SELECT id FROM customers WHERE customer_hash='<hash>');
```

5. **NICHT löschen**:
   - `orders.id`, `total_cents`, `created_at`, `legal_retention_until` — Steuerrechtsschutz
   - `order_items` — Order-Statistik
6. **Bestätigungs-Email** an Kunde senden mit Erklärung was gelöscht wurde + was 10y bleibt

---

## 7 · SITE OFFLINE NEHMEN (NOTFALL)

**Use-Case**: Datenleck, fehlerhaftes Pricing live, Cyber-Attack.

### Variante A — Pre-Launch-Gate aktivieren (sanft)

1. `gate.js` ist heute aktiv. Wenn live → Wiederherstellen:

```
ssh root@<VPS-IP>
cd /root/blackdress
git checkout HEAD~1 -- gate.js  # oder aus backups/ kopieren
bash /root/deploy_cloudflare.sh
```

### Variante B — Cloudflare-Pages-Disable (sofort, am wirksamsten)

1. https://dash.cloudflare.com → Pages → blackdress-demo
2. Settings → Pause deployments
3. Site bleibt aktuelle Cache-Inhalte servieren bis CDN-TTL abläuft (max 1h)
4. Alternative: Custom Domain entfernen → www.blackdress.ch zeigt CF-Default-Page

### Variante C — DNS-Switch (totale Kontrolle)

1. Wix-Account (Registrar) → DNS-Settings
2. CNAME `www` von `blackdress-demo.pages.dev` → temporäre Maintenance-Page
3. **Achtung**: Mail-DNS (MX, DKIM, TXT) NICHT ändern!

---

## 8 · BACKUP WIEDERHERSTELLEN

### `products.json` aus Backup

1. SSH zum VPS
2. 
```
cd /root/blackdress
ls -la backups/                  # zeige verfügbare Snapshots
cp data/products.json backups/products_pre_restore_$(date +%Y-%m-%d-%H%M).json
cp backups/products_pre_<gewählter-snapshot>.json data/products.json
bash /root/deploy_cloudflare.sh
```

### Hostinger VPS-Snapshot (komplette Disk)

1. https://hpanel.hostinger.com → VPS → Snapshots
2. Snapshot wählen → Restore
3. **Achtung**: VPS ist während Restore down (~10-30 Min). Site-Cache läuft weiter aber neue Genesis-Aktionen sind blockiert
4. Nach Restore: Genesis-Service prüfen `systemctl status claude`

### D1-Datenbank

- (Sobald deployed) `wrangler d1 export blackdress-prod --output=backup.sql`
- Restore: `wrangler d1 execute blackdress-prod --file=backup.sql`

---

## 9 · VPS NEU STARTEN (HOSTINGER-PANEL)

1. https://hpanel.hostinger.com → VPS → `srv1611647`

2. Power-Menü → "Reboot"
3. Wartezeit: ~30 Sekunden
4. Verify: SSH-Connection (sobald wieder up):

```
ssh root@<VPS-IP>
uptime    # erwartet: <1 min seit reboot
```

5. Genesis-Service: `systemctl status claude` (sollte automatisch starten via systemd-Service)

**Achtung**: Cloudflare-Pages + DNS sind extern (NICHT auf VPS) → Site bleibt während VPS-Reboot weiter erreichbar.

---

## 10 · GENESIS NEU STARTEN

```
ssh root@<VPS-IP>
systemctl restart claude
```

**Wartezeit**: ~5 Sekunden.

**Verify**:

1. Telegram-Test-Nachricht senden an `@Genesis1608_Bot` (Don's Chat)
2. Genesis sollte innerhalb 30 Sek antworten

**Wenn Genesis nach Restart NICHT antwortet**: siehe §11.

---

## 11 · DIAGNOSE: GENESIS ANTWORTET NICHT

### Symptome

- Telegram-Nachricht an @Genesis1608_Bot bleibt unbeantwortet
- Don sendet message → keine Reaction → keine Reply

### Schritt-für-Schritt

1. **systemd-Status prüfen**:

```
ssh root@<VPS-IP>
systemctl status claude
```

   - **Active (running)** → weiter zu Schritt 2
   - **Failed** → `journalctl -u claude --since "1h" | tail -50` für Error-Log

2. **Memory-Check** (häufigste Ursache: OOM):

```
free -h
```

   - Wenn `available` <100 MB → OOM-Risiko, Genesis evtl. gekillt
   - Fix: Swap aktivieren (sollte schon sein per 2026-05-08 Setup); falls nicht: `swapon --show`

3. **Telegram-Bot-Service**:

```
ps aux | grep -i telegram
cat /root/.claude/channels/telegram/.env  # prüfe Token vorhanden
```

4. **Restart-Versuch** (siehe §10): `systemctl restart claude`
5. **Wenn weiter nicht antwortet**: Reboot ganzes VPS (siehe §9)
6. **Fallback**: Don schreibt sich selbst per Email → später wenn Genesis online die Rückstand-Liste durchgehen

---

## 12 · DIAGNOSE: VPS DOWN

### Symptome

- SSH-Connect-Refused
- www.blackdress.ch funktioniert eventuell weiter (CF-Pages-Cache!)

### Schritt-für-Schritt

1. https://hpanel.hostinger.com → VPS-Status
   - **Running, IP erreichbar** → SSH-Issue (Schritt 2)
   - **Stopped/Crashed** → Power → Start
   - **Maintenance** → warten bis Hostinger-Wartung durch
2. **Network-Diagnose**:

```
ping <VPS-IP>           # erreicht IP?
curl -I http://<VPS-IP>:22   # SSH-Port offen?
```

3. **Hostinger-VNC-Console** (wenn SSH dead aber VPS running):
   - Hostinger-Panel → VPS → "Browser SSH"
   - Direkt-Login als root → diagnostizieren
4. **Logs**:

```
journalctl --since "30 min ago" | tail -100
dmesg | tail -50    # Kernel-Errors, OOM
```

5. **Letzter Ausweg**: Hostinger-Snapshot zurückspielen (siehe §8)

---

## 13 · DIAGNOSE: DOMAIN NICHT ERREICHBAR

### Symptome

- www.blackdress.ch zeigt nichts / Error / "Site nicht gefunden"

### Schritt-für-Schritt

1. **DNS-Resolution prüfen**:

```
dig www.blackdress.ch
```

   - **Antwort mit IP** → DNS ist OK, Problem liegt bei Cloudflare-Pages
   - **Keine Antwort / NXDOMAIN** → DNS-Issue (Schritt 2)

2. **Wix-DNS-Status**:
   - https://wix.com → Login → Domain-Settings → blackdress.ch
   - Verify CNAME `www` → `blackdress-demo.pages.dev`
   - Verify A-Record für `@` (apex) → `188.114.97.3` (CF-Pages-IP)
   - **NICHT die MX/TXT/DKIM ändern** (Mail-Hosting)!

3. **Cloudflare-Pages-Custom-Domain**:
   - https://dash.cloudflare.com → Pages → blackdress-demo → Custom Domains
   - Verify `www.blackdress.ch` ist mit Status "Active"
   - Falls "Pending" → einige Minuten warten + nochmal prüfen

4. **SSL-Issue**:

```
curl -vI https://www.blackdress.ch  2>&1 | head -30
```

   - SSL-Errors → Cloudflare-Universal-SSL prüfen (Dashboard → SSL/TLS → Edge Certificates)

5. **Reverse-Test**: Direkter Pages-URL versuchen

```
curl -I https://blackdress-demo.pages.dev
```

   - Funktioniert → Problem ist nur Custom-Domain-Routing
   - Funktioniert nicht → Pages-Site selbst down (CF-Status-Page checken: https://www.cloudflarestatus.com)

---

## 14 · OG-IMAGE / BRAND-ASSET AUSTAUSCHEN

**Use-Case**: Logo-Refresh, Saison-Banner, Re-Branding.

### Schritt-für-Schritt

1. SSH zum VPS:

```
ssh root@<VPS-IP>
cd /root/blackdress/assets
```

2. **Neues OG-Image erzeugen** (1200×630, JPEG, ~70 KB optimal):
   - Externes Tool (Photoshop/Figma/Canva) → Export als `og-blackdress-v2.jpg`
   - ODER PIL-Script aus `/tmp/build_og_image.py` als Vorlage anpassen

3. **Datei-Upload via SCP**:

```
scp og-blackdress-v2.jpg root@<VPS-IP>:/root/blackdress/assets/
```

4. **Bulk-Update aller HTML-Pages** auf neuen Filename:

```
cd /root/blackdress
sed -i 's|og-blackdress\.jpg|og-blackdress-v2.jpg|g' *.html blog/*.html
```

5. **Verify**: `grep -l 'og-blackdress' *.html | head` → alle sollten neuen Filename zeigen
6. **Deploy**:

```
bash /root/deploy_cloudflare.sh
```

7. **Cache-Bust**: Neuer Filename ist automatischer Telegram-Cache-Bust (Telegram caches per-URL).
8. **Test**: Beliebigen blackdress.ch-Link via Telegram-Share → neue Vorschau erscheint sofort

### Falls Filename gleich bleiben soll (z.B. nur Pixel-Edit)

- Cloudflare Pages cached aggressive (`Cache-Control: max-age=31536000, immutable` per `_headers`)
- Lösung: Cloudflare-Dashboard → Pages → blackdress-demo → Caching → Purge by URL
- Telegram cached ebenfalls aggressive — kein Server-Side-Purge möglich, nur via neuen Filename

---

## 15 · CACHE-HEADERS (_HEADERS) ANPASSEN

**Use-Case**: Cache-TTL ändern, neue Asset-Typen, Security-Header-Update.

### Datei-Pfad: `/root/blackdress/_headers`

### Aktuelles Profil (Step P, Stand 2026-05-08)

| Pattern | Cache-Control | Begründung |
|---|---|---|
| `/assets/*.{jpg,png,webp,svg,jpeg}` | `max-age=31536000, immutable` (1 Jahr) | Bilder ändern sich nie ohne neuen Filename |
| `/assets/*.css` + `/assets/*.js` + `/lib/*.js` | `max-age=2592000` (30 Tage) | Re-Deploy invalidiert via filename-Hash |
| `/gate.js` + `/cookie-banner.js` + `/sw.js` | `max-age=2592000` (30 Tage) | Root-level JS |
| `/data/*.json` | `max-age=300, must-revalidate` (5 Min) | Stock kann sich ändern |
| `/*.html` und `/` | `max-age=0, must-revalidate` (immer fresh) | Inhaltliche Änderungen |
| `/woff2`, `/woff` | `max-age=31536000, immutable` | Fonts ändern sich nie |

### Security-Headers (für `/*` site-wide)

- `X-Content-Type-Options: nosniff`
- `X-Frame-Options: SAMEORIGIN`
- `Referrer-Policy: strict-origin-when-cross-origin`
- `Permissions-Policy: geolocation=(), microphone=(), camera=()`

### Anpassung

1. SSH + Edit `/root/blackdress/_headers`
2. Syntax-Check: pattern-line, dann TAB-eingerückte Header-Zeilen, Leerzeile als Block-Trenner
3. Deploy: `bash /root/deploy_cloudflare.sh`
4. Verify: `curl -sLI https://www.blackdress.ch/<URL> | grep -i cache-control`

---

## 16 · CSS / JS BEARBEITEN (SOURCE + RE-BUILD)

**Wichtig**: Die produktiven Files (style.css, app.js, ...) sind **minified**. Bei Edits IMMER über die `*.src`-Source-Files arbeiten.

### Schritt-für-Schritt

1. SSH + zum Source-File:

```
ssh root@<VPS-IP>
cd /root/blackdress/assets
nano style.css.src      # oder app.js.src, etc.
```

2. Edit speichern.
3. **Re-Build** (idempotent):

```
cd /root/blackdress
python3 build_minify.py
```

   Output zeigt vorher/nachher-Größen aller 6 Files.
4. **Deploy**:

```
bash /root/deploy_cloudflare.sh
```

### Source-Files-Liste

| Source | Generated |
|---|---|
| `assets/style.css.src` | `assets/style.css` |
| `assets/app.js.src` | `assets/app.js` |
| `cookie-banner.js.src` | `cookie-banner.js` |
| `gate.js.src` | `gate.js` |

| | |
|---|---|
| `lib/aes.js.src` | `lib/aes.js` |
| `sw.js.src` | `sw.js` |

### Verify nach Deploy

```
curl -s https://www.blackdress.ch/assets/style.css | head -3
# erwartet: minified-Output (alles in einer Zeile, kein Whitespace)
```

### Notfall-Rollback

- `cp /root/blackdress/assets/style.css.src /root/blackdress/assets/style.css` (skipt minify, deployed unminified)
- ODER: Git-Revert wenn Repo unter Version-Control kommt (TODO Phase F)

---

## 17 · COOKIE-CONSENT FÜR USER RESET

**Use-Case**: Kunde meldet "Cookie-Banner kommt nicht mehr" oder "ich will neu wählen".

### User-seitige Lösung (Don's Antwort an Kunde)

"Bitte besuche `https://www.blackdress.ch/cookie-einstellungen` — dort findest du den Button 'Einwilligung widerrufen'. Beim nächsten Seitenbesuch wird der Banner erneut angezeigt."

### Server-seitige Hilfe (sollte nicht nötig sein, aber dokumentiert)

- Cookie-Consent ist 100% client-side (LocalStorage-Key `bd_cookie_consent_v1`)
- Server kennt diese Wahl NICHT — kann nicht remote-resetten
- Bei Bug-Reports: Browser-Console + LocalStorage-Inspect (`localStorage.getItem('bd_cookie_consent_v1')`)

### Cookie-Banner-Bug-Diagnose

1. Browser-Console öffnen → Tab "Application" → LocalStorage
2. Key `bd_cookie_consent_v1` checken
3. Wenn fehlend: Banner sollte beim Reload erscheinen
4. Wenn vorhanden aber falsch: `BD_Cookie.clear()` in Console aufrufen → Reload

---

## 18 · EMAIL-WORKER DEPLOYEN (TEST → LIVE)

**Vorbereitung**: Worker-Skelett ist seit Step R ready in `/root/blackdress/workers/email/`. Provider-Wahl pendend (Don entscheidet: Brevo vs CF Email-Workers).

### Schritt-für-Schritt

```
ssh root@<VPS-IP>
cd /root/blackdress/workers/email
npm install                                  # nur erste Mal
```

### Test-Mode-Deploy

```
set -a && source /root/.cloudflare && set +a       # CF-Token laden
wrangler login                                     # falls noch nicht
wrangler secret put WEBHOOK_AUTH_TOKEN             # generiere: openssl rand -hex 32
wrangler secret put BLACKDRESS_AES_KEY             # 64-char-hex AES-256 key
wrangler secret put EMAIL_PROVIDER                 # → 'stub' für Erst-Test
wrangler deploy
```

Output: `https://blackdress-email.<account>.workers.dev`

### Smoke-Test

```
TOKEN="<WEBHOOK_AUTH_TOKEN-Wert>"
curl https://blackdress-email.<account>.workers.dev/health
curl -X POST https://blackdress-email.<account>.workers.dev/send/order-confirmation
\
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"order_number":"BD-2026-00001"}'
```

Mit `EMAIL_PROVIDER=stub` wird KEIN Email versendet, aber email_log-Eintrag in D1 geschrieben (Test ohne Provider-Quota-Burn).

### Switch to Brevo (live)

```
wrangler secret put EMAIL_PROVIDER       # → 'brevo'
wrangler secret put BREVO_API_KEY        # <key from brevo.com>
wrangler deploy
```

Voraussetzung: Domain `blackdress.ch` als verified Sender in brevo.com Dashboard registrieren + DKIM/SPF-CNAMEs einrichten.

### Switch to Production-D1

```
# Nach blackdress-prod-Migration:
# Edit wrangler.toml [env.production.d1_databases] section mit prod-DB-ID
wrangler deploy --env production
```

### Trigger-Wiring (Payrexx-Webhook)

Payrexx-Dashboard → Webhooks → URL: `https://blackdress-email.<account>.workers.dev/webhook/order-status` Methode: POST · Header: `Authorization: Bearer <WEBHOOK_AUTH_TOKEN>` Body: `{ "order_number": "BD-...", "new_status": "paid" }`

---

## 19 · E2E TEST-SUITE LOKAL LAUFEN

**Test-Suite**: 13/13 Cases grün (Suite 1 Browse+Cart + Suite 9 Security-Basics) seit Step Z 2026-05-08. Restliche 45 Cases (Suite 2-8) sind als test.skip() bis D1-Prod + Email-Worker + Payrexx live.

### Erst-Setup

```
ssh root@<VPS-IP>
cd /root/blackdress/test_plans/playwright
npm install -D @playwright/test typescript      # falls erste Mal
npx playwright install-deps chromium            # System-Libs
npx playwright install chromium                 # Browser ~112 MB
```

### Run aktive Tests

```
cd /root/blackdress/test_plans/playwright
npx playwright test specs/01-browse-cart.spec.ts specs/09-security-basics.spec.ts
```

Erwarteter Output: `13 passed (~22s)`. Tests laufen gegen www.blackdress.ch mit gate.js+age-gate+cookie-consent automatisch bypassed via helpers.

### Bei Failure

1. `cat test-results/*/error-context.md` für Detail
2. `npx playwright show-trace test-results/*/trace.zip` für Visual-Debug
3. Nicht test.skip()-zurück (Don's Disziplin) — diagnostizieren + fixen ODER als known-issue ins Brain

### Test-Schreib-Pattern (für künftige Suites)

- Helpers in `helpers/index.ts` nutzen (bypassGate, addToCart, clearCart, customerHash, aesEncrypt, applyDiscountWithCap)
- Schema 1:1 spiegeln (CART_SCHEMA = 2 number, NICHT 'v2' string!)
- ensureOnSite() vor localStorage-Access (sonst SecurityError on about:blank)

---

## 20 · PAGES-INDEX SCHNELLZUGANG

**Use-Case**: QA-Klick-Test, Live-Verify, Fehler-Korrektur — schneller Zugriff auf alle 72 Pages.

### Drei Wege:

1. **Dashboard-Sektion**: bot-imperium.pages.dev → Section "🗂 BlackDress · Pages-Index" (klickbar, Tab-target=_blank)
2. **Markdown-File**: `/root/blackdress/manual/blackdress_pages_index.md` (alle URLs + Speziell-URLs für Test-Produkte/Sitemap/OG-Image)
3. **PDF-Download**: bei Operations-Manual-Sektion 4. Card

### Pre-Launch-Hinweis

Alle URLs benötigen aktiven gate.js-Bypass. Im Browser einmal Pre-Launch-PW eingeben → LocalStorage merkt sich Token (`bd_gate_v1=1`) → persistent für Session.

### Bei Bug auf einer Page:

1. Klick aus Pages-Index → Browser-DevTools öffnen
2. Console-Tab für JS-Errors

3. Network-Tab für 404 / 500-Responses
4. Findings an Genesis (Telegram) → Fix folgt

---

## 21 · WO IM ZWEIFEL FRAGEN

| Problem | Anlaufstelle |
|---|---|
| Hostinger-spezifisch | https://www.hostinger.com/contact (Live-Chat 24/7) |
| Cloudflare-spezifisch | https://www.cloudflarestatus.com + https://community.cloudflare.com |
| Payrexx | https://www.payrexx.com/de/support |
| DSGVO/Recht | Anwalt (Don's Vertrauen) |
| OG-Image / Telegram-Vorschau-Cache | Neuer Filename = automatischer Cache-Bust |
| CSS/JS Minify-Bug | `*.src` ist Source-of-Truth → editieren + re-build |
| Cookie-Banner UX | https://www.blackdress.ch/cookie-einstellungen → Reset-Button |
| Anything else | Genesis (sobald online) — Telegram an @Genesis1608_Bot |

---

## Update-Disziplin

Diese Datei wird aktualisiert:

- Bei neuem Operations-Workflow (z.B. wenn cmo-bot live → Newsletter-Send-Anleitung)
- Bei Architektur-Änderung die Don-Aktionen verändert
- Pre-Live: Don liest komplett durch + gibt Feedback → Genesis update bis verständlich

---

**Cross-Refs**:

- `blackdress_zugangs_inventar.md` — alle Credentials + Speicherorte
- `blackdress_architektur.md` — Datenfluss + Tabellen-Schema + Crons