combo-raport-app

Jak zalogowa? si? do Arcane (przewodnik dla adminów)

Jak zalogowa? si? do Arcane (przewodnik dla adminów)

Adres Arcane: https://arcane.t-pizza.pl **Adres Authentika (SSO):** https://id.t-pizza.pl

Masz **dwa sposoby** logowania:

1. **Logowanie lokalne** (login + hasło bezpośrednio w Arcane) — działa zawsze, niezależnie od Authentika
2. **Logowanie przez Authentik (SSO/OIDC)** — wygodniejsze, jedno konto dla wielu usług w przyszłości

Strona logowania pokazuje **oba przyciski jednocześnie**: zwykły formularz login/password + przycisk „Sign in with Authentik".

Scenariusz A: Istniej?cy admin (Twój login ju? jest w Arcane)

Dotyczy: `admin-combo@t-pizza.pl`, `kamil.lendlewicz@telepizza.pl`, `matzxp84@gmail.com` — kontá utworzone przed wdrożeniem SSO.

Wariant A1 — najprostszy: zaloguj si? jak dawniej (lokalne has?o)

1. Otwórz https://arcane.t-pizza.pl
2. Wpisz **swój login** (email) + **lokalne hasło Arcane**
3. Kliknij **Sign in**
4. Gotowe — żaden Authentik nie jest potrzebny

To samo, co przed wdrożeniem SSO. Twoje hasło ani konto nie zostało zmienione.

Wariant A2 — przejd? na SSO (zalecane na przysz?o??)

Wymaga: konta w Authentiku z **tym samym emailem** co konto w Arcane.

Krok 1 — admin Authentika (m.fleis) utworzy ci konto:

- Otwiera https://id.t-pizza.pl/if/admin/ → Directory → Users → Create
- Username: `kamil.lendlewicz` (część przed `@`)
- Email: `kamil.lendlewicz@telepizza.pl` ← **MUSI być identyczny** z emailem w Arcane (bez tego konta się nie skleją)
- Name: imię i nazwisko
- Path: `users`
- Klika **Create**
- Następnie w detalach usera klika **Set Password** lub **Email password reset link**
- Dodaje cię do grupy `arcane-admin` (Directory → Groups → arcane-admin → Members → Add)

Krok 2 — Ty się logujesz:

1. Otwierasz https://arcane.t-pizza.pl
2. Klikasz **Sign in with Authentik**
3. Authentik prosi o login/hasło → wpisujesz swoje dane z Authentika
4. (Jeśli pierwszy raz logujesz się do Arcane przez SSO) Authentik pokaże ekran „Authorize" — kliknij **Authorize**
5. Wracasz do Arcane jako zalogowany admin

Co się stało pod spodem: Arcane dostał z Authentika twój email `kamil.lendlewicz@telepizza.pl`. Sprawdził lokalną bazę → znalazł istniejące konto z tym emailem → skleił konta (`oidcMergeAccounts=true`). Od tego momentu możesz logować się oboma sposobami — to ten sam user.

Co je?li zapomnia?em lokalnego has?a Arcane?

Arcane nie ma self-service password reset. Inny admin musi zresetować:

- Inny admin loguje się do Arcane → Settings → Users → znajdź usera → **Reset password**
- Albo: wszyscy adminowie z dostępem do VPS-a mogą zresetować via API (technical procedure dla zaufanych)

Alternatywa: jeśli admin Authentika utworzy ci konto z tym samym emailem, możesz zalogować się przez SSO (nie potrzebując lokalnego hasła) — następnie w Arcane → Profile → Set password

ustaw nowe lokalne hasło.

Scenariusz B: Nowy admin (nikt go nie ma w Arcane)

Dotyczy: ktoś nowy w organizacji, kto nie ma jeszcze konta w Arcane.

Wymagania

- Konto w Authentik (zakłada admin Authentika, m.fleis)
- Członkostwo w grupie `arcane-admin` w Authentik (claim który Arcane mapuje na rolę admin)

Procedura dla admina Authentika

1. Otwórz https://id.t-pizza.pl/if/admin/ → **Directory → Users → Create**
2. Wypełnij:
   - **Username:** np. `j.kowalski` (część przed `@`)
   - **Email:** `j.kowalski@telepizza.pl` (pełny email)
   - **Name:** Jan Kowalski
   - **Path:** `users`
3. Kliknij **Create**
4. W detalach świeżo utworzonego usera kliknij ⋮ → **Set password** (wpisz tymczasowe hasło i przekaż userowi przez bezpieczny kanał) **LUB** ⋮ → **Email password reset link** (system wyśle link na email — wymaga SMTP, mamy skonfigurowany)
5. **Directory → Groups →** `arcane-admin` **→ Users → Add existing user** → wybierz `j.kowalski` → Add

Procedura dla nowego admina (Jan Kowalski)

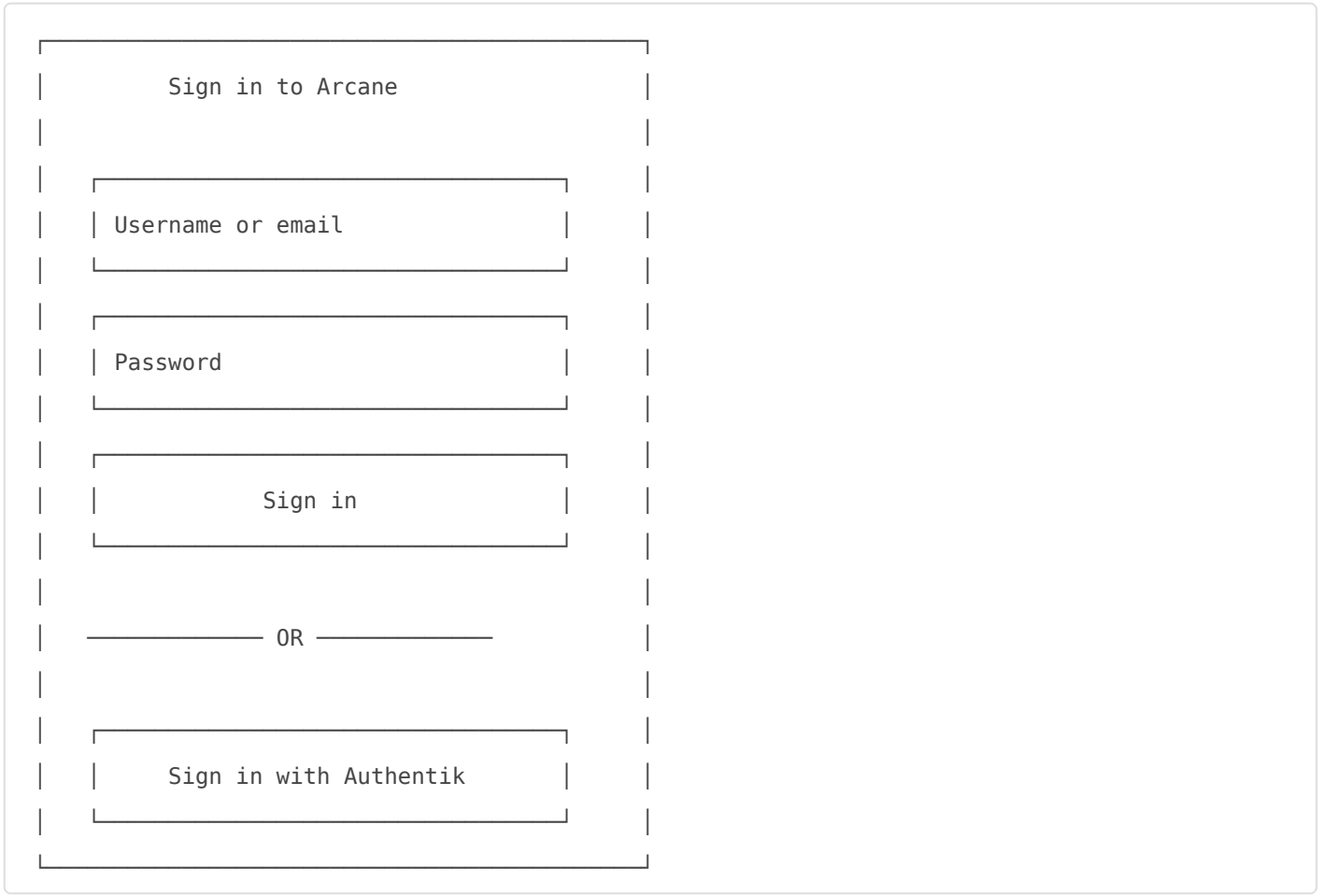
1. Sprawdź skrzynkę `j.kowalski@telepizza.pl` — jeśli admin Authentika wysłał link resetu hasła, kliknij go i ustaw swoje hasło
2. Otwórz https://arcane.t-pizza.pl
3. Kliknij **Sign in with Authentik**
4. Authentik prosi o login/hasło → wpisz `j.kowalski` (lub `j.kowalski@telepizza.pl`) + hasło ustawione w kroku 1
5. (Pierwszy raz) Kliknij **Authorize** na ekranie consent
6. Wracasz do Arcane — **konto zostało utworzone automatycznie** (Arcane nie miał takiego usera, więc go utworzył przy pierwszym SSO login)

7. Sprawdź: w prawym górnym rogu Arcane powinieneś widzieć swój email + ikonkę admin

Od teraz logujesz się tylko przez SSO. Twoje hasło w Arcane nie istnieje (chyba że je sobie wyklikasz w Profile → Set password — to dodatkowe lokalne hasło dla scenariusza break-glass).

Co wida? na stronie logowania

Strona `https://arcane.t-pizza.pl` (gdy nie jesteś zalogowany) wygląda tak:



- **Górna część** — logowanie lokalne (login/hasło bezpośrednio w Arcane). Działa dla `admin-combo`, `kamil.lendlewicz`, `matzxp84`.
- **Dolny przycisk** — przekierowanie do Authentika. Po loginie w Authentiku wracasz tu zalogowany.

Mapowanie ról i dost?p do aplikacji

W Authentiku jest **policy binding** na aplikacji `Arcane`: tylko członkowie grupy **`arcane-admin`** widzą i mogą używać aplikacji. Jeśli ktoś zaloguje się do Authentika ale nie jest w `arcane-admin`, na ekranie

consent dostanie błąd "permission denied" / "Application is not accessible to this user".

Wewnątrz Arcane: claim `groups` z ID tokenu mapuje się na rolę:

Twoja grupa w Authentik	Co widzisz w Arcane (po SSO)
`arcane-admin`	**admin** (pełne uprawnienia, widzisz aplikację, możesz zarządzać kontenerami)
inna / żadna	**dostęp odrzucony przez policy** — nie zobaczysz aplikacji w Authentiku

Wniosek: żeby ktoś mógł zalogować się do Arcane przez SSO, **musi być w grupie** `arcane-admin` **w Authentiku**. Nie ma „zwykłego usera SSO" — albo admin, albo nie wpuszczamy.

Dla istniejących adminów lokalnych (`admin-combo`, `kamil.lendlewicz`, `matzxp84`): ich rola admin jest zapisana lokalnie w Arcane — działa niezależnie od Authentika. Po sklejeniu kont (mergeAccounts) zachowują rolę admin.

FAQ

P: Co jeśli Authentik padnie / nie odpowiada? O: Strona logowania Arcane nadal pokazuje formularz lokalny — możesz zalogować się hasłem bezpośrednio w Arcane (jeśli masz lokalne konto). Authentik jest opcjonalną drogą, nie obowiązkową.

P: Mogę zalogować się raz przez SSO, a innym razem lokalnym hasłem? O: Tak, jeśli masz oba: konto w Authentiku **i** lokalne hasło w Arcane. Konta są sklejone, więc to ten sam user — ale możesz wybrać sposób logowania za każdym razem.

P: Czy SSO obejmuje też combo.t-pizza.pl? O: Nie, na razie tylko Arcane. Combo-raport-app ma własny system użytkowników (lokalny). W przyszłości można dodać OIDC client do `combo-raport-app` i wpiąć tę samą aplikację w Authentiku.

P: Jak zmienić swoje hasło w Authentiku? O: https://id.t-pizza.pl/if/user/ → User settings (ikona w prawym górnym rogu) → Change password.

P: Jak zmienić swoje hasło lokalne w Arcane? O: Po zalogowaniu w Arcane → Profile (prawy górny róg) → Set/Change password. Tylko lokalnie — nie wpływa na Authentika.

P: Co znaczy „mergeAccounts" w praktyce? O: Authentik wysyła email do Arcane. Arcane szuka usera z tym emailem:

- **Znalazł** lokalnego → ten sam user, dodaje połączenie z OIDC (od teraz oba sposoby logowania).

- **Nie znalazł** → tworzy nowego usera (tylko OIDC, bez lokalnego hasła do czasu aż user sam je ustawi).

P: Jak admin Authentika może zobaczyć kto się zalogował? O: https://id.t-pizza.pl/if/admin/ → Events → wszystkie eventy login/logout/authorize.

P: Czy hasło z Authentika jest takie samo jak z Arcane? O: **Nie.** To dwa oddzielne hasła (chyba że celowo ustawisz takie same). Authentik trzyma swoje hasło w swojej bazie; Arcane trzyma swoje lokalne hasło osobno. SSO oznacza tylko że Arcane *ufa* Authentikowi że potwierdził tożsamość — nie wymienia haseł.

Lista obecnych userów (stan na 2026-05-28)

W Arcane (lokalni adminowie)

Email	Username	Rola	Konto w Authentik?
`admin-combo@t-pizza.pl`	admin-combo@t-pizza.pl	admin	**tak** (utworzone 2026-05-28, gotowe do SSO)
`kamil.lendlewicz@telepizza.pl`	kamil.lendlewicz@telepizza.pl	admin	**tak** (utworzone 2026-05-28, gotowe do SSO)
`matzxp84@gmail.com`	matzxp84	admin	**tak** (utworzone 2026-05-28, gotowe do SSO)
`m.fleis@czcyber.pl`	akadmin	admin	**tak** (zlane z Authentik akadmin, super-admin Authentika)

W Authentik (wszyscy w grupie `arcane-admin`)

Username	Email	Grupy	Hasło startowe
`akadmin`	m.fleis@czcyber.pl	authentik Admins, arcane-admin	ustawione wcześniej
`admin-combo`	admin-combo@t-pizza.pl	arcane-admin	wygenerowane, zapisane w `~/.secrets/authentik-users.env`
`kamil.lendlewicz`	kamil.lendlewicz@telepizza.pl	arcane-admin	wygenerowane, zapisane w `~/.secrets/authentik-users.env`

Username	Email	Grupy	Hasło startowe
`matzxp84`	matzxp84@gmail.com	arcane-admin	wygenerowane, zapisane w `~/.secrets/authentik-users.env`

Przy pierwszym logowaniu SSO: user wpisuje swoje hasło startowe w Authentiku → Arcane mergeAccounts wykrywa pasujący email → konto w Arcane staje się dostępne także przez SSO (lokalne hasło Arcane dalej działa równolegle).

Zalecane: każdy user po pierwszym loginie powinien zmienić hasło: https://id.t-pizza.pl/if/user/ → User settings → Change password.

Policy binding (kto widzi aplikacj? Arcane)

Aplikacja `Arcane` w Authentiku ma policy binding (pk=`4a4d46af-...`, enabled, order=0): wymaga członkostwa w grupie `arcane-admin`. Jeśli ktoś spoza tej grupy zaloguje się do Authentika i kliknie aplikację Arcane → dostanie błąd "permission denied".

To znaczy: żeby kogoś dopuścić do Arcane przez SSO, **wystarczy dodać go do grupy `arcane-admin`** w Authentiku (Directory → Groups → arcane-admin → Users → Add existing user). Nie trzeba nic robić po stronie Arcane — wszystko dzieje się przez claim `groups` w ID tokenie.

Dla adminów Authentika — quick reference

Utwórz usera: Directory → Users → Create **Dodaj do grupy:** Directory → Groups → `arcane-admin` → Users → Add existing **Reset hasła:** Directory → Users → wybierz usera → ⋮ → Set password / Email password reset link **Wyłącz usera:** Directory → Users → wybierz usera → ⋮ → Set inactive (zostaje w bazie, ale nie może się logować) **Usuń usera:** Directory → Users → wybierz usera → ⋮ → Delete (UWAGA: nieodwracalne) **Eventy login:** Events → Logs (filtruj po username) **Sesje aktywne:** Directory → Users → wybierz usera → tab "Sessions" → Revoke

Break-glass (jeśli `akadmin` zgubi hasło):

```
ssh root@212.132.103.157
docker exec authentik-worker ak shell -c "
from authentik.core.models import User
u = User.objects.get(username='akadmin')
u.set_password('NOWE-HASLO-TUTAJ')
u.save()
```

```
print('OK')
"
```

Authentik ? Arcane SSO (post-implementation)

Authentik ? Arcane SSO (post-implementation)

Status: wdrożone 2026-05-27/28 **VPS:** home.pl, 212.132.103.157 (Debian 13 trixie)
Architektura: Authentik (OIDC provider) → Arcane v1.19.5 (relying party), bez forward-auth/outpost

---

1. Topologia

```
Public Internet (212.132.103.157)
        |
   nginx :80/:443  (Debian host)
        ├─ combo.t-pizza.pl   → 127.0.0.1:4173 ──┐
        ├─ arcane.t-pizza.pl  → 127.0.0.1:3552 ──┤
        └─ id.t-pizza.pl      → 127.0.0.1:9000 ──┤
                                                 |
                        docker network "web" (bridge, external)
                        ├─ combo_prod          (combo-raport-prod:latest)
                        ├─ arcane              (ghcr.io/getarcaneapp/arcane:latest)
                        ├─ authentik-server    (ghcr.io/goauthentik/server:2026.2.3)
                        ├─ authentik-worker    (ghcr.io/goauthentik/server:2026.2.3)
                        ├─ authentik-postgresql (postgres:16-alpine)
                        └─ authentik-redis     (redis:alpine)
```

Arcane → Authentik server-side: po nazwie kontenera w sieci `web` (autodiscovery `/.well-known/openid-configuration`). Przeglądarka usera → Authentik: przez publiczny `https://id.t-pizza.pl/`.

---

2. Stack Authentik

Lokalizacja: `/opt/docker/authentik/`

- `compose.yaml` — server + worker + postgresql + redis, healthcheck `ak healthcheck`, depends_on `service_healthy`
- `.env` (chmod 600) — `PG_PASS`, `AUTHENTIK_SECRET_KEY`, SMTP, tag obrazu

Tag: `AUTHENTIK_TAG=2026.2.3` (stable; nie `latest`)

Volumes (named):

- `authentik_authentik-postgres` — baza danych (kluczowy dla recovery)
- `authentik_authentik-redis` — sesje/cache
- `authentik_authentik-media` — media (loga, favicony brand)
- `authentik_authentik-templates` — custom email/HTML templates
- `authentik_authentik-certs` — embedded outpost certs

SMTP: `serwer2104579.home.pl:587` STARTTLS, sender `authentik@t-pizza.pl` (skrzynka shared host home.pl).

Ekspozycja: `127.0.0.1:9000:9000` — tylko loopback, TLS terminuje nginx.

---

3. nginx vhost

`/etc/nginx/sites-available/id.t-pizza.pl` (symlink w `sites-enabled/`):

- Listen 443 ssl (ECDSA cert Let's Encrypt, auto-renew certbot.timer)
- Listen 80 → 301 redirect na HTTPS (managed by Certbot)
- `proxy_pass http://127.0.0.1:9000`
- `Upgrade` / `Connection upgrade` (WebSocket dla flow embed Authentika)
- `proxy_read_timeout 86400` (long polling)
- X-Forwarded-Host/Proto/For — KLUCZOWE, bez tego Authentik składa złe URL-e

Cert: `Certificate Name: id.t-pizza.pl`, ECDSA, expiry 2026-08-25 (auto-renew).

---

4. Konfiguracja Authentik (przez API)

Zbootstrapowane przez `https://id.t-pizza.pl/api/v3/` z API token akadmin:

Provider OAuth2/OpenID (pk=1):

- Name: `arcane-oidc`
- Client type: confidential
- Client ID: `arcane`
- Client secret: 128 znaków (zapisany w `~/.secrets/arcane-oidc.env` chmod 600)
- Authorization flow: `default-provider-authorization-implicit-consent`
- Invalidation flow: `default-provider-invalidation-flow`
- Signing key: `authentik Self-signed Certificate`
- Scopes: `openid` + `profile` + `email`
- Sub mode: `hashed_user_id`, Issuer mode: `per_provider`
- Redirect URIs (strict): `https://arcane.t-pizza.pl/auth/oidc/callback`

Application: `Arcane` (slug `arcane`, provider pk=1, launch URL `https://arcane.t-pizza.pl`)

Group: `arcane-admin` (non-superuser) — member: `akadmin`

Issuer URL: `https://id.t-pizza.pl/application/o/arcane/` **Discovery:** `https://id.t-pizza.pl/application/o/arcane/.well-known/openid-configuration`

---

5. Konfiguracja Arcane (przez API, settings w DB)

Arcane v1.19.5 trzyma OIDC w DB settings — modyfikujesz przez `PUT /api/environments/0/settings` (X-API-Key), bez recreate kontenera.

Kluczowe ustawienia:

Key	Value	Komentarz
`baseServerUrl`	`https://arcane.t-pizza.pl`	KRYTYCZNE — bez tego redirect URI źle składany
`oidcEnabled`	`true`	
`oidcClientId`	`arcane`	
`oidcClientSecret`	(128 znaków)	
`oidcIssuerUrl`	`https://id.t-pizza.pl/application/o/arcane/`	autodiscovery `/.well-known/openid-configuration`
`oidcScopes`	`openid profile email`	
`oidcAdminClaim`	`groups`	mapowanie ról: claim z ID token
`oidcAdminValue`	`arcane-admin`	wartość claim = admin w Arcane
`oidcProviderName`	`Authentik`	label na buttonie login

Key	Value	Komentarz
`oidcMergeAccounts`	`true`	scalanie kont po emailu (bez tego: duplicate user)
`oidcAutoRedirectToProvider`	`false`	**break-glass** — login page nadal pokazuje lokalny form
`authLocalEnabled`	`true`	lokalny admin Arcane nadal aktywny

---

6. Backup

`/opt/docker/arcane/backup-cron.sh` (uprzednio istniejący) iteruje po **wszystkich woluminach Docker** via Arcane Volume Backup API (`POST /api/environments/0/volumes/{name}/backups`). Działa też dla `authentik_*` volumes — bez modyfikacji.

Cron: `30 3 * * *` (root) — codziennie 03:30. **Log:** `/var/log/arcane-backup.log` **Rotacja:** max 10 backupów/wolumin (skrypt usuwa najstarsze). **Lokalizacja backupów:** `/opt/backups/arcane/` (bind mount → Arcane `/backups`).

Backupy off-site (manualny, do disaster recovery):

```
rsync -avz root@212.132.103.157:/opt/backups/arcane/ ~/backups/vps-home-pl/
```

---

7. Break-glass procedura

Jeśli Authentik padnie / OIDC nie działa:

1. **Lokalny admin Arcane:** otwórz `https://arcane.t-pizza.pl/`, kliknij „local login" (przycisk widoczny, bo `oidcAutoRedirectToProvider=false`), zaloguj się jako `admin-combo@t-pizza.pl` (lokalne konto + hasło z password managera) — pełen dostęp do UI niezależnie od Authentika.
2. **Wyłączyć OIDC bez restartu** (jeśli trzeba):
   ```
   curl -X PUT -H "X-API-Key: $ARCANE_KEY" -H "Content-Type: application/json" \
     https://arcane.t-pizza.pl/api/environments/0/settings \
     -d '{"oidcEnabled":"false"}'
   ```
3. **Authentik direct admin:** `https://id.t-pizza.pl/if/admin/` jako `akadmin` (hasło w `~/.secrets/authentik-akadmin.env`, plus break-glass przez `docker exec authentik-worker ak shell` jeśli kompletny lockout).

---

8. Rollback (gdyby co? posz?o ?le)

Pre-flight backup z 2026-05-27 23:17 leży w:

- VPS: `/opt/backups/pre-authentik-20260527-211758/`
- Off-site lokalnie: `~/backups/vps-home-pl/pre-authentik-20260527-211758/`

Zawiera: tarbale `/opt/docker`, `/etc/nginx`, `/etc/letsencrypt`, kopię `docker-compose.yml` combo, manifesty docker (ps/images/networks/volumes), iptables.rules.

Volume backups Arcane sprzed wdrożenia: `arcane_arcane-data-1779916672384855682-688ba987`, `combo_data-1779916673232502773-41a003ba` (restore via `POST /api/environments/0/volumes/backups/{id}/restore`).

---

9. ?cie?ki/komendy do zapami?tania

```
# Status wszystkich stacks
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Status}}"

# Logi Authentika
docker logs authentik-server --tail 100
docker logs authentik-worker --tail 100

# Logi Arcane
docker logs arcane --tail 100

# Recreate Authentika (po zmianie compose/.env)
cd /opt/docker/authentik && docker compose up -d

# Sprawdzenie Authentik discovery
curl -s https://id.t-pizza.pl/application/o/arcane/.well-known/openid-configuration | jq

# Sprawdzenie status OIDC w Arcane (publiczne, bez auth)
curl -s https://arcane.t-pizza.pl/api/oidc/status | jq

# Backup manualny (uruchom skrypt)
/opt/docker/arcane/backup-cron.sh
tail -f /var/log/arcane-backup.log
```

Backup

Backup

Automatyczny backup danych produkcyjnych na **Google Drive** via `rclone`.

Co jest backupowane?

Wolumin Docker	Zawartość	Kontener
`combo_data`	`auth.json`, `email.json`	`combo_prod` (port 4173)
`combo_data_87`	`auth.json`, `email.json`	`combo_prod87` (port 87)

> Kod źródłowy jest już chroniony przez git (GitHub). Backup dotyczy wyłącznie danych runtime — kont użytkowników i konfiguracji emaila.

Jednorazowy setup

1. Instalacja rclone

```
sudo apt install rclone
# lub najnowsza wersja:
curl https://rclone.org/install.sh | sudo bash
```

Weryfikacja:

```
rclone --version
```

2. Konfiguracja Google Drive

```
rclone config
```

Przejdź przez kreator:

```
n  → New remote
Name: gdrive
Storage: drive        (wpisz numer odpowiadający "Google Drive")
client_id:            (Enter — zostaw puste)
client_secret:        (Enter — zostaw puste)
scope: 1              (Full access)
root_folder_id:       (Enter — zostaw puste)
service_account_file: (Enter — zostaw puste)
Edit advanced config: n
Use auto config: y    → otworzy przeglądarkę, zaloguj się na konto Google
Configure as a Shared Drive: n
y  → potwierdzenie
q  → wyjście
```

Weryfikacja dostępu:

```
rclone lsd gdrive:
```

Powinny pojawić się foldery z Twojego Google Drive.

3. Test manualny

```
cd /home/ultrax/Projects/combo-project/combo-raport-app
./scripts/backup-gdrive.sh
```

Po wykonaniu sprawdź Google Drive — powinien pojawić się folder `combo-raport-backup/`.

Automatyczny backup — cron

Konfiguracja

Dodaj wpis do crontab:

```
crontab -e
```

Wklej linię (backup codziennie o **03:00**):

```
0 3 * * * /home/ultrax/Projects/combo-project/combo-raport-app/scripts/backup-gdrive.sh >>
/var/log/combo-backup.log 2>&1
```

Zapisz i wyjdź. Weryfikacja:

```
crontab -l
```

Sprawdzanie logów

```
# Ostatnie 50 linii logu
tail -50 /var/log/combo-backup.log

# Podgląd na żywo
tail -f /var/log/combo-backup.log
```

Skrypt backup-gdrive.sh — zachowanie

- Tworzy archiwum `tar.gz` z zawartości każdego wolumenu
- Upload do `gdrive:combo-raport-backup/<nazwa_wolumenu>/`
- Usuwa pliki na GDrive starsze niż **30 dni**
- Pomija wolumin jeśli nie istnieje (np. `combo_data_87` przed pierwszym uruchomieniem prod87)
- Loguje każdy krok z timestampem

Uruchomienie manualne

```
# Backup oba wolumeny
./scripts/backup-gdrive.sh

# Backup tylko jednego wolumenu
./scripts/backup-gdrive.sh combo_data
./scripts/backup-gdrive.sh combo_data_87
```

Struktura na Google Drive

```
combo-raport-backup/
├── combo_data/
│   ├── combo_data_20260415_030001.tar.gz
│   ├── combo_data_20260416_030001.tar.gz
│   └── ...
└── combo_data_87/
    ├── combo_data_87_20260415_030001.tar.gz
    └── ...
```

Odtworzenie danych (disaster recovery)

W przypadku utraty środowiska:

```
# 1. Pobierz najnowszy backup z Google Drive
rclone copy gdrive:combo-raport-backup/combo_data/ ./restore-tmp/ --include "*.tar.gz"

# 2. Sprawdź pobrane pliki
ls -lh ./restore-tmp/

# 3. Odtwórz wolumin (zastępuje obecną zawartość!)
ARCHIVE=$(ls ./restore-tmp/*.tar.gz | sort | tail -1)
docker run --rm \
  -v combo_data:/data \
  -v "$(pwd)/restore-tmp:/backup:ro" \
  alpine \
  sh -c "cd /data && tar xzf /backup/$(basename ${ARCHIVE})"

# 4. Uruchom aplikację
docker compose --profile prod up -d

# 5. Usuń tymczasowy katalog
rm -rf ./restore-tmp/
```

Zmiana cz?stotliwo?ci backupu

Przykładowe harmonogramy cron:

Harmonogram	Wyrażenie cron
Codziennie o 03:00	`0 3 * * *`
Co 12 godzin	`0 3,15 * * *`
Co poniedziałek o 02:00	`0 2 * * 1`

Edycja:

```
crontab -e
```

Zmiana okresu przechowywania

W pliku `scripts/backup-gdrive.sh` zmień wartość `KEEP_DAYS`:

```
KEEP_DAYS=30   # domyślnie 30 dni
```

Dane raportowe

Dane raportowe

Wszystkie dane raportowe są statycznymi plikami JSON w katalogu `public/data/`. Vite serwuje je bezpośrednio — brak backendu, brak API.

Konwencja nazewnictwa plików

```
public/data/T{tableId}/T{tableId}L{listId}{locationId}.json
```

Przykład: `T1L12830.json` → tabela **T1**, lista **L1**, lokalizacja **2830**

Listy (list_id)

Wyświetlana nazwa	list_id	Przykładowy plik
Rafał Lubak	L1	`T1L12830.json`
Rafał Wieczorek	L2	`T1L22830.json`
Andrzej Chmielewski	L3	`T1L32830.json`

W selektorze UI wyświetlany jest format: `Rafał Lubak (L1)`. Kod mapuje nazwę → `list_id` przy budowaniu ścieżki do pliku.

Sekcje raportu (table_id)

table_id	Sekcja	Format danych
T1	Informacje o wolumenie miesięcznym	`ReportRow[]`
T2	Kluczowe wskaźniki miesięczne (KPI)	`KpiRow[]`
T5	Sprzedaż od początku roku (YTD)	`ReportRow[]`

Każda sekcja w DOM posiada atrybut `data-table-id="T1"` — przydatne przy automatyzacji lub scrapingu.

Format JSON — T1 i T5 (`ReportRow[]`)

```
[
  {
    "id": "2026",
    "label": "2026",
    "cells": [
      { "value": "88 045" },
      { "value": "79 546" },
      { "value": "X", "highlight": true },
      { "value": "196 424", "highlightBg": true }
    ]
  }
]
```

Pola `CellValue`

Pole	Typ	Opis
`value`	`string`	Wyświetlana wartość komórki
`highlight`	`boolean?`	Tekst w kolorze amber (`text-amber-500`)
`highlightBg`	`boolean?`	Tło amber (`bg-amber-500/15`) + `highlight`

Format JSON — T2 (`KpiRow[]`)

```
[
  {
    "id": "avg-sales",
    "label": "Średnia sprzedaż",
    "cells": ["1 234", "1 100", "980", "..."]
  }
```

```
]
```

Kolejność komórek odpowiada kolejności kolumn miesięcznych (od najnowszego).

Wska?niki T2 (kolejno?? A–K)

ID wiersza	Etykieta	Typ wartości
`avg-sales`	Średnia sprzedaż	zł
`customers-count`	Ilość klientów	liczba
`other-sales-qty`	Sprzedaż pozostałe	liczba
`customers-yoy`	Klienci vs rok poprzedni %	%
`sales-pizza-total`	Pizza	zł
`pizzas-yoy`	Pizze vs rok poprzedni %	%
`drinks-sales`	Napoje	zł
`drinks-pct`	Sprzedaż napoje %	%
`addons-sales`	Sprzedaż dodatków	zł
`starters-sales`	Startery	zł
`avg-bill`	Średni rachunek	zł

Wiersze pieniężne (`zł`) mają automatycznie doklejony sufiks `zł` w UI.

Mapowanie ID komórek — T1

Wiersze (lata)

Etykieta w JSON	ID w UI
`2026`	`TY`
`2025`	`LY`
`2024`	`AY`
`2026 vs 2025`	`VS1`
`2025 vs 2024`	`VS2`

Kolumny miesi?czne — aliasy specjalne (TY)

Miesiąc	ID
`TY` + `02` (luty)	`TYLM`
`TY` + `03` (marzec)	`TYTM`
`TY` + `04` (kwiecień)	`TYNM`

Pozostałe kolumny: `{yearAlias}{monthId}` → np. `LY03`

Przyk?adowe ID komórek T1

Komórka	ID
TY, luty	`TYLM`
TY, marzec	`TYTM`
LY, styczeń	`LY01`
VS1, czerwiec	`VS106`

Mapowanie ID komórek — T2

Kolumny mają techniczne ID miesięczne:

Miesiąc	ID kolumny
Mar 2026	`TM` (bieżący miesiąc)
Lut 2026	`M1`
Sty 2026	`M2`
Gru 2025	`M3`
Lis 2025	`M4`
Paź 2025	`M5`
Wrz 2025	`M6`
Sie 2025	`M7`
Lip 2025	`M8`
Cze 2025	`M9`
Maj 2025	`M10`
Kwi 2025	`M11`
Mar 2025	`MR`

Wiersze mają ID literowe Excel: `A`, `B`, ..., `K`.

Format ID komórki T2: `{litera}-{miesiacId}` → np. `A-TM`, `B-M1`, `K-MR`

Dodanie nowych danych

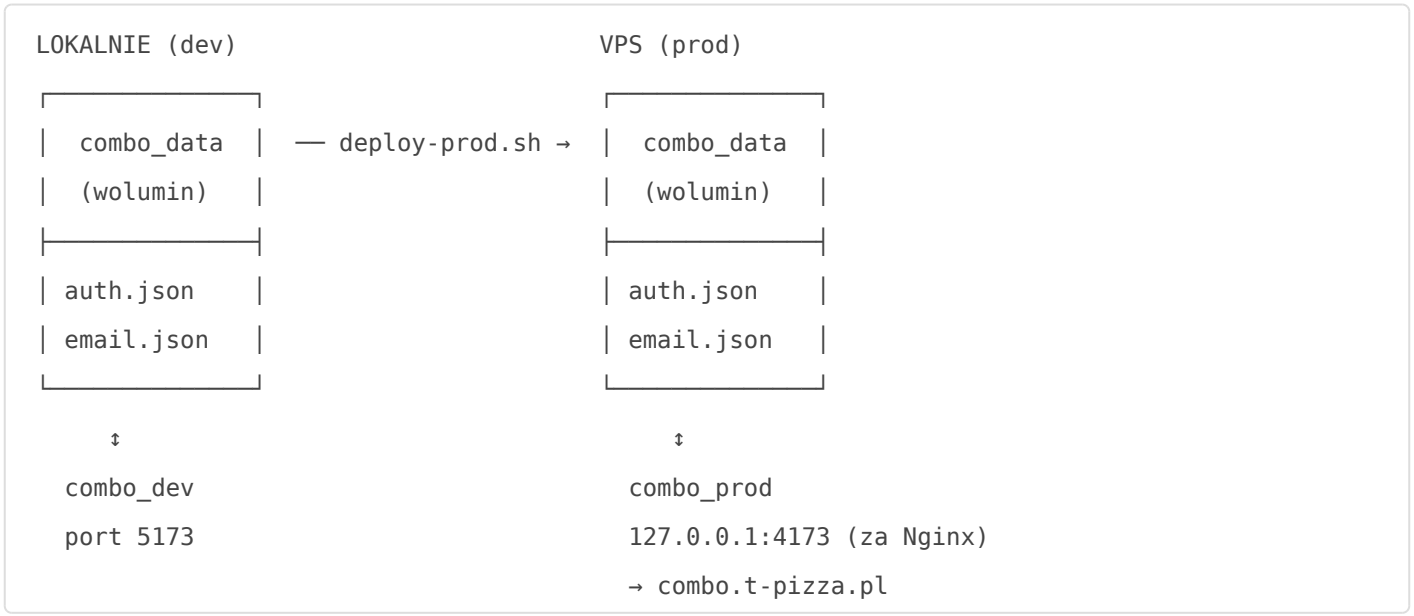
1. Utwórz plik JSON w odpowiednim katalogu (`public/data/T1/`, `T2/`, `T5/`)
2. Nazwij plik zgodnie z konwencją: `T1L{listId}{locationId}.json`
3. Jeśli dodajesz nową listę — zarejestruj ją w selektorze w `src/App.tsx`
4. Zapisz plik — Vite wykryje zmianę i automatycznie przeładuje stronę (bez restartu kontenera)

Deploy na produkcj?

Deploy na produkcj?

Skrypt `scripts/deploy-prod.sh` automatyzuje przenoszenie aplikacji + danych z lokalnego środowiska dev na serwer produkcyjny (home.pl ComboVPS → `combo.t-pizza.pl`).

Architektura danych



Plik	Zawartość
`auth.json`	Użytkownicy, hasła (scrypt), JWT secret, logi audytowe
`email.json`	SMTP config, harmonogramy raportów, szablony email, lista odbiorców

---

Jednorazowy setup

1. Skonfiguruj po??czenie SSH

Klucz SSH do VPS (`root@212.132.103.157`). Sugerowany wpis w `~/.ssh/config`:

```
Host vps
    HostName 212.132.103.157
    User root
    IdentityFile ~/.ssh/jprdl
```

2. Utwórz plik `.deploy.env`

```
cp .deploy.env.example .deploy.env
```

Domyślna konfiguracja (host=vps, klucz=~/.ssh/jprdl, katalog=/opt/docker/combo-raport, profil=prod) działa bez zmian, jeśli używasz wpisu `vps` z `~/.ssh/config`.

```
DEPLOY_HOST=vps
DEPLOY_SSH_PORT=22
DEPLOY_USER=root
DEPLOY_SSH_KEY=~/.ssh/jprdl
DEPLOY_APP_DIR=/opt/docker/combo-raport
DEPLOY_PROFILE=prod
DEPLOY_CONTAINER=combo_prod
```

> `.deploy.env` jest w `.gitignore` — nigdy nie trafi do repozytorium.

3. Przygotuj VPS

Na serwerze (jednorazowo):

```
# Sieć Docker `web` (external) — utworzona w trakcie VPS init
docker network ls | grep web

# Klon repo
mkdir -p /opt/docker
git clone git@github.com:matzxp84/combo-raport-app.git /opt/docker/combo-raport

# Vhost Nginx → docker/nginx-vhost.conf
cp /opt/docker/combo-raport/docker/nginx-vhost.conf \
   /etc/nginx/sites-available/combo.t-pizza.pl.conf
ln -s /etc/nginx/sites-available/combo.t-pizza.pl.conf /etc/nginx/sites-enabled/
nginx -t && systemctl reload nginx
```

```
# SSL (po wskazaniu DNS A → 212.132.103.157)
certbot --nginx -d combo.t-pizza.pl
```

U?ycie

Pe?ny deploy (tylko kod, domy?lnie)

```
./scripts/deploy-prod.sh
```

Co robi:

1. Kopiuje `.env` (secrets GoPos) na serwer przez SCP
2. Na VPS: `git pull --ff-only` + `docker compose --profile prod up -d --build`
3. Sprawdza healthcheck kontenera `combo_prod`

Kod + synchronizacja danych

```
./scripts/deploy-prod.sh --with-data
```

Dodatkowo:

1. Eksportuje `auth.json` i `email.json` z lokalnego wolumenu `combo_data`
2. Kopiuje je na serwer przez SCP
3. Importuje do wolumenu `combo_data` na VPS

Tylko dane

```
./scripts/deploy-prod.sh --data-only
```

Przydatne gdy dodałeś użytkowników w panelu admina na dev i chcesz ich przenieść na prod.

One-shot: commit + push + deploy

```
./scripts/publish.sh "commit message"
# lub
pnpm publish:prod
```

Co jest przenoszone

Element	Jak przenoszone	Gdzie ląduje
Kod źródłowy	`git pull --ff-only`	`/opt/docker/combo-raport/`
`.env` (GoPos secrets)	SCP	`/opt/docker/combo-raport/.env`
`auth.json` (użytkownicy)	Docker volume export/import	wolumin `combo_data`
`email.json` (SMTP, szablony)	Docker volume export/import	wolumin `combo_data`

Co NIE jest przenoszone

- `node_modules` — instalowane na serwerze podczas `docker build`
- `dist/` — budowane na serwerze podczas `docker build`
- Logi Docker — per kontener, nie migrowane

Secrets — bezpiecze?stwo

Secret	Plik	Ochrona
`GOPOS_CLIENT_ID` / `GOPOS_CLIENT_SECRET`	`.env`	`.gitignore` + `.dockerignore`
JWT secret	`auth.json` (w wolumenie)	nie w repo, w Docker volume
Hasła użytkowników	`auth.json` (scrypt hash)	hashowane, nie w repo
Hasło SMTP	`email.json` (w wolumenie)	nie w repo, w Docker volume

Zasady:

- `.env` i `.deploy.env` NIGDY nie trafiają do gita
- Dane runtime (`auth.json`, `email.json`) żyją w Docker volumes, nie w repo
- Backup danych: `scripts/backup-gdrive.sh` (patrz Backup)
- Przy pierwszym uruchomieniu bez danych serwer tworzy seed users — **zmień hasła natychmiast**

Seed users (domy?lne)

Email	Hasło	Rola
`matfl@tuta.com`	`pułtusk`	admin

Email	Hasło	Rola
`daniel.piekarski@t-pizza.pl`	`daniel`	user

> Tworzone TYLKO gdy `auth.json` jest pusty (brak użytkowników). Jeśli deployujesz z danymi — seed nie występuje.

Troubleshooting

Nie mog? po??czy? si? z serwerem

```
# Test ręczny
ssh vps "echo ok"
# lub bezpośrednio
ssh -i ~/.ssh/jprdl root@212.132.103.157 "echo ok"
```

Sprawdź: klucz SSH, wpis w `~/.ssh/config`, czy IP nie jest zbanowane w fail2ban (`fail2ban-client status sshd`).

Kontener nie startuje

```
ssh vps "docker logs combo_prod --tail 50"
```

Brak miejsca na dysku

Przed deployem wyczyść dysk na serwerze:

```
ssh vps "docker system prune -a && apt clean"
```

Dane nie przenios?y si?

Sprawdź zawartość wolumenu na serwerze:

```
ssh vps "docker run --rm -v combo_data:/data alpine cat /data/auth.json"
```

Nginx / SSL

```
ssh vps "nginx -t && systemctl reload nginx"
ssh vps "certbot certificates"
```

Docker

Docker

Projekt używa **Docker Desktop** z wieloetapowym `Dockerfile` i `docker-compose.yml` opartym na profilach.

Profile

Profil	Kontener	Port	Opis
`dev`	`combo_dev`	`5173`	Vite hot-reload, pliki montowane z hosta
`prod`	`combo_prod`	`4173`	nginx serwujący zbudowany `dist/`

Komendy

Uruchomienie — tryb dev

```
docker compose --profile dev up
```

Aplikacja dostępna pod **http://localhost:5173**

Każda zmiana w `src/` lub `public/data/` jest widoczna w przeglądarce natychmiast bez restartu kontenera.

Uruchomienie — tryb prod

```
docker compose --profile prod up --build
```

Aplikacja dostępna pod **http://localhost:4173**

Zatrzymanie

```
docker compose --profile dev down
# lub
docker compose --profile prod down
```

Rebuild po zmianie zale?no?ci (`package.json`)

```
docker compose --profile dev build --no-cache
docker compose --profile dev up
```

Podgl?d logów (live)

```
docker logs combo_dev -f
```

Shell wewn?trz kontenera

```
docker exec -it combo_dev sh
```

Sie?

Projekt używa izolowanej sieci `combo_net` (bridge, `172.23.0.0/16`).

Nie koliduje z istniejącymi kontenerami WordPress:

Kontener	Port	Sieć
`spzw_wp`	8080	`spzw_frontend`
`spzw_pma`	8081	`spzw_frontend`
`spzw_db`	3306	`spzw_backend`
`combo_dev`	**5173**	`combo_net`
`combo_prod`	**4173**	`combo_net`

Struktura plików Docker

```
combo-raport-app/
├── Dockerfile          # 5 etapów: base → deps → dev / build → prod
├── docker-compose.yml  # profile dev i prod
├── .dockerignore       # wyklucza node_modules, dist, .git, .claude
└── docker/
    └── nginx.conf      # SPA routing, cache assetów, nagłówki bezpieczeństwa
```

Dockerfile — etapy

```
base  (node:22-alpine + pnpm via corepack)
 └─ deps  (pnpm install --frozen-lockfile, cache mount)
     ├─ dev    → EXPOSE 5173, CMD vite --host 0.0.0.0
     ├─ build  → RUN pnpm build
     └─ prod   → nginx:1.27-alpine, EXPOSE 4173
```

Etap `prod` kopiuje tylko `dist/` — nie zawiera `node_modules` ani kodu źródłowego.

---

nginx.conf — kluczowe zachowania

- **SPA routing** — wszystkie ścieżki przekierowywane do `index.html` (`try_files $uri /index.html`)
- **Cache assetów** — JS/CSS/fonty/obrazy: `Cache-Control: public, immutable, 1 rok`
- **Brak cache dla index.html** — `no-cache, no-store, must-revalidate`
- **Kompresja gzip** — włączona dla JS, CSS, JSON, SVG
- **Nagłówki bezpieczeństwa** — `X-Frame-Options`, `X-Content-Type-Options`, `Referrer-Policy`

Wskazówki

Zmiana portu dev — edytuj `docker-compose.yml` (`"5173:5173"`) i dodaj `server: { port: XXXX }` w `vite.config.ts`, następnie zrób rebuild.

Logi w Docker Desktop — zakładka *Containers* → kliknij `combo_dev` → zakładka *Logs*.

Uruchomienie bez Dockera (lokalnie) — patrz Struktura projektu → Skrypty pnpm.

combo-raport-app — Wiki

combo-raport-app — Wiki

Interaktywna aplikacja raportowa do wizualizacji danych sprzedażowych i wolumenowych z systemu GoPos. Dane pobierane przez API, raporty wysyłane automatycznie emailem.

Interaktywna aplikacja raportowa do wizualizacji miesięcznych danych sprzedażowych i wolumenowych. Dane ładowane dynamicznie z plików JSON według wybranej listy i miesiąca.

Spis tre?ci

Szybki start

```
# Tryb developerski (hot-reload)
docker compose --profile dev up
# → http://localhost:5173

# Produkcja (nginx)
docker compose --profile prod up --build
# → http://localhost:4173
```

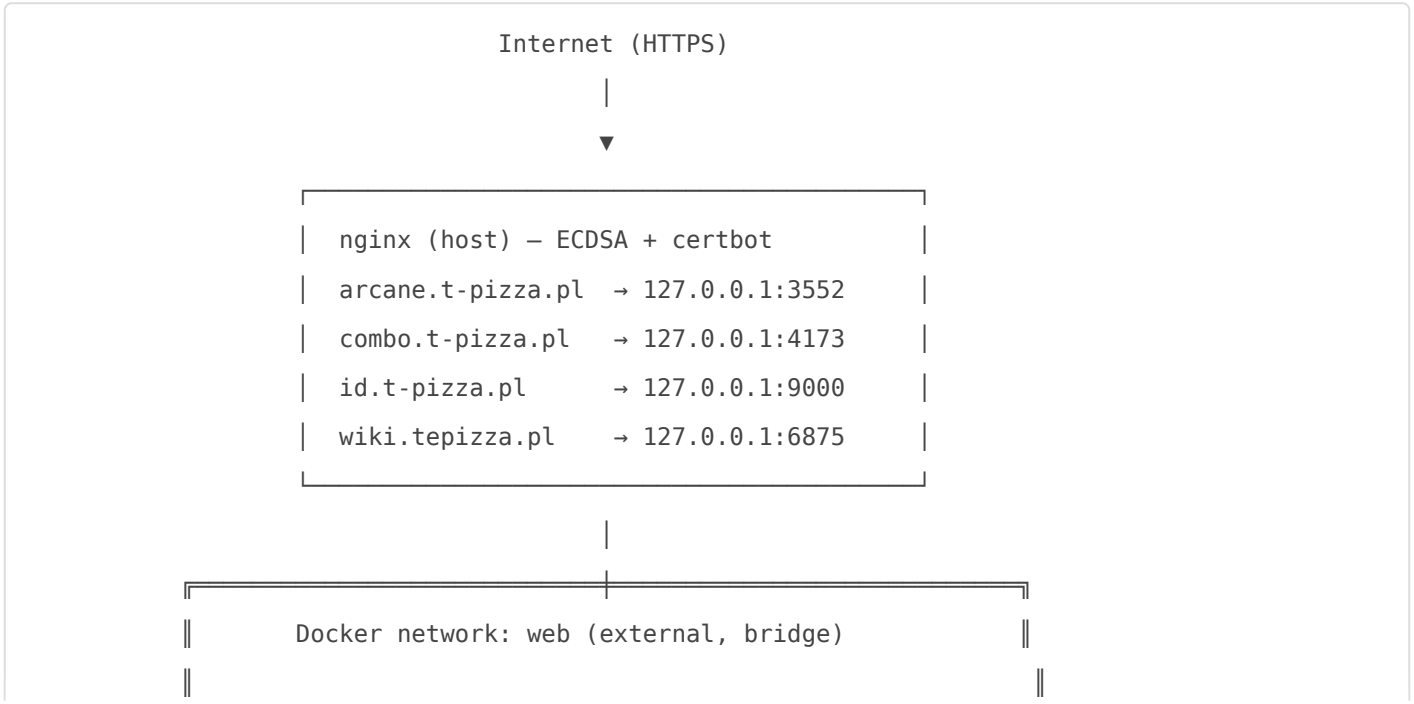
Stack

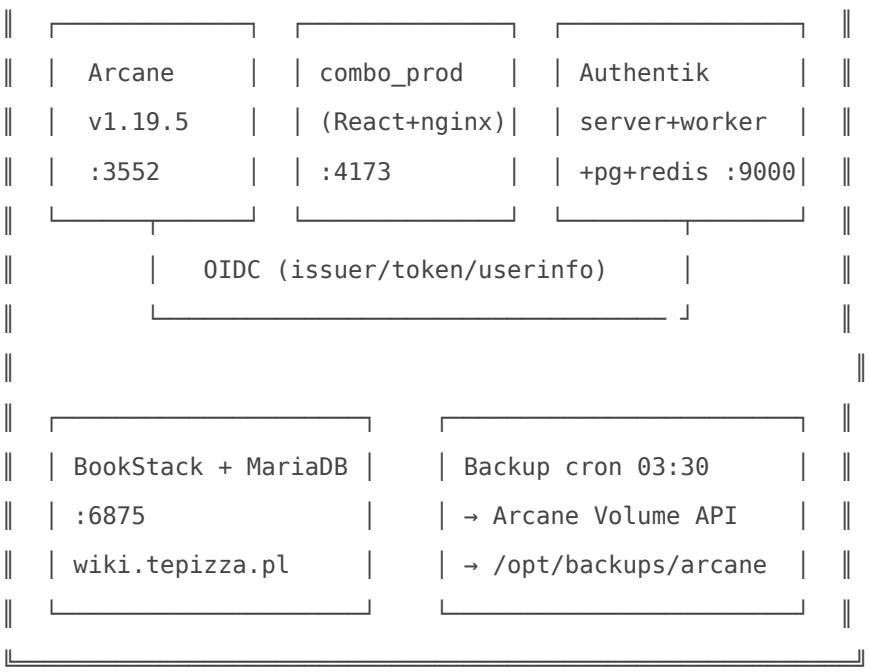
Warstwa	Technologia
UI	React 19, Tailwind CSS 4, shadcn/ui, Base UI
Tabele	TanStack Table v8
Build	Vite 7, TypeScript 5.9
Kontener	Docker multi-stage, nginx 1.27-alpine

Schemat ?rodowiska produkcyjnego

> **VPS home.pl** · `212.132.103.157` · Debian 13 trixie · 4 vCPU / 7.7 GiB RAM / 237 GiB SSD
> **Edytowalny diagram drawio:** plik `env-prod.drawio` w sekcji *Attachments* pod tą stroną — pobierz, otwórz na https://app.diagrams.net lub w VS Code (Draw.io Integration), zedytuj i wgraj z powrotem.

Topologia (widok logiczny)





Stacki Docker

Compose	Kontener	Port (host)	Domena	Notatki
`/opt/docker/arcane/compose.yaml`	`arcane`	`127.0.0.1:3552`	`arcane.t-pizza.pl`	Arcane v1.19.5 — OIDC w DB settings
`/root/combo-raport-app/docker-compose.yml`	`combo_prod`	`127.0.0.1:4173`	`combo.t-pizza.pl`	React 19 + nginx-alpine, volume `combo_data`
`/opt/docker/authentik/compose.yaml`	`authentik-{server,worker,postgresql,redis}`	`127.0.0.1:9000`	`id.t-pizza.pl`	Authentik 2026.2.3, SMTP via `serwer2104579.home.pl:587`
`/app/data/projects/bookstack/compose.yaml` *(Arcane-managed)*	`bookstack`, `bookstack-db`	`127.0.0.1:6875`	`wiki.tepizza.pl`	BookStack 26.3.5 + MariaDB 11, `client_max_body_size 50M`

SSO / OIDC

- **Provider:** Authentik (`id.t-pizza.pl`) — app `arcane` (pk=1) prod, `arcane-dev` (pk=3) lokalny
- **Relying party:** Arcane v1.19.5 — konfiguracja w DB (nie env), zmiany bez recreate kontenera
- **Redirect URI:** `https://arcane.t-pizza.pl/auth/oidc/callback` *(uwaga: frontend route, nie `/api/...`)*
- **Admin claim:** `groups` → grupa `arcane-admin` (członkowie: `akadmin`, `admin-combo`, `kamil.lendlewicz`, `matzxp84`)

- **Break-glass:** lokalny login Arcane pozostaje włączony (`authLocalEnabled=true`, `oidcAutoRedirectToProvider=false`)

Backup

- **Skrypt:** `/opt/docker/arcane/backup-cron.sh` — iteruje po **wszystkich** wolumenach Docker via Arcane Volume Backup API
- **Cron:** `30 3 * * *` (codziennie 03:30, przed logrotate)
- **Log:** `/var/log/arcane-backup.log`
- **Lokalizacja:** `/opt/backups/arcane/` (bind-mount jako `/backups` w kontenerze Arcane), rotacja **10** kopii na wolumen
- **Off-site:** `rsync -avz root@212.132.103.157:/opt/backups/arcane/ ~/backups/vps-home-pl/`

Backupowane wolumeny: `arcane_arcane-data`, `combo_data`, `authentik_authentik-{postgres,redis,media,templates,certs}`.

Jak zedytowa? ten schemat

1. Otwórz tę stronę w trybie edycji (BookStack → *Edit*).
2. **Tekst / tabele** — wszystko powyżej jest zwykłym markdown/HTML, edytuj bezpośrednio w WYSIWYG.
3. **Diagram graficzny** — pobierz `env-prod.drawio` z sekcji *Attachments*, otwórz w https://app.diagrams.net, zapisz lokalnie, wgraj ponownie jako attachment (nadpisze stary).
4. Po zmianie infrastruktury zaktualizuj tabelę *Stacki Docker* oraz topologię ASCII.

Komponenty

Komponenty

Architektura UI opiera się na komponentach shadcn/ui (headless + Tailwind), rozszerzonych o własną logikę raportową.

Hierarchia komponentów

```
main.tsx
└── ThemeProvider          # kontekst motywu (dark/light/system)
    └── TooltipProvider    # globalny provider tooltipów (Radix)
        └── App            # główna logika: selektor listy/miesiąca, sekcje raportu
            ├── DarkModeToggle
            ├── ReportTable (T1)
            ├── KpiMonthlyTable (T2)
            └── ReportTable (T5)
```

ThemeProvider

Plik: `src/components/theme-provider.tsx`

Zarządza motywem całej aplikacji.

API

```
// Odczyt i zmiana motywu w dowolnym komponencie:
const { theme, setTheme } = useTheme()

setTheme("dark")    // zawsze ciemny
setTheme("light")   // zawsze jasny
setTheme("system")  // podążaj za OS
```

Zachowania

- Motyw persystowany w `localStorage` (klucz: `"theme"`)
- Przy starcie odczytuje zapisaną wartość; brak → `"system"`
- Reaguje na zmianę `prefers-color-scheme` w czasie rzeczywistym (gdy `theme === "system"`)
- Reaguje na `StorageEvent` — synchronizacja między zakładkami
- **Skrót klawiszowy** `D` — przełącza dark/light bez użycia myszki (działa gdy focus nie jest na polu tekstowym)
- Przełączenie motywu tymczasowo wyłącza CSS transitions (brak migania)

DarkModeToggle

Komponent `Switch` powiązany z `useTheme`. Wyświetla aktualny stan i pozwala go przełączyć kliknięciem.

ReportTable (T1, T5)

Tabela wolumenowa/YTD renderowana przez **TanStack Table v8**.

Props / dane wej?ciowe

- Dane: `ReportRow[]` ładowane z pliku JSON (lub hardcoded `reportData` w T1)
- Kolumny generowane dynamicznie na podstawie miesięcy (z `MONTHS`)
- Każda komórka może mieć `highlight` (kolor tekstu) lub `highlightBg` (tło)

Funkcje ID

Funkcja	Opis
`getDisplayRowId(label, fallback)`	Mapuje rok na alias: `2026→TY`, `2025→LY`, `2024→AY`, `vs→VS1/VS2`
`getBaseYearTwoDigits(label, fallback)`	Zwraca dwuznakowy prefiks dla ID komórki
`getT1MonthCellId(base, monthId)`	Buduje ID komórki: `TY+02→TYLM`, `TY+03→TYTM`, `TY+04→TYNM`

Atrybuty DOM

Atrybut	Przykład	Opis
`data-table-id`	`data-table-id="T1"`	Na kontenerze sekcji

Atrybut	Przykład	Opis
`data-row-id`	`data-row-id="TY"`	Na wierszu
`data-cell-id`	`data-cell-id="TYLM"`	Na komórce

KpiMonthlyTable (T2)

Tabela KPI z 13 stałymi kolumnami miesięcznymi i 11 wskaźnikami.

Kolumny miesi?czne

Zdefiniowane jako stała `KPI_MONTH_COLUMNS` — od Mar 2026 wstecz do Mar 2025. Kolumna `Mar 2026` (bieżący miesiąc) wyświetla zawsze wartość `TM` (placeholder — dane live).

Props `KpiMonthlyTable`

Prop	Typ	Domyślnie	Opis
`data`	`KpiRow[]`	`kpiMonthlyData`	Dane tabeli
`showIds`	`boolean`	`false`	Pokazuje techniczne ID kolumn i wierszy
`hidePercent`	`boolean`	`false`	Ukrywa znak `%` w wartościach
`hidePln`	`boolean`	`false`	Ukrywa sufiks `zł`

KpiLabelCell

Komponent renderujący etykietę wskaźnika z kolorowaniem składni:

- **tekst główny** — `font-medium`
- **(opis w nawiasach)** — kolor `text-syntax-opisy`
- **[slug w nawiasach kwadratowych]** — `font-mono`, kolor `text-syntax-slug`

CellContent

Wspólny komponent komórki dla T1/T5:

```
<CellContent cell={{ value: "88 045", highlight: true }} hidePercent={false} />
```

- `highlight: true` → `text-amber-500 dark:text-amber-400`
- `highlightBg: true` → `bg-amber-500/15 dark:bg-amber-500/20 rounded px-1`
- `hidePercent: true` → usuwa znak `%` z wartości

Komponenty UI (shadcn)

Wszystkie w `src/components/ui/`. Dodane przez CLI `npx shadcn@latest add <nazwa>`.

Plik	Komponent	Użycie
`table.tsx`	`Table`, `TableRow`, `TableCell`, `TableHead`, `TableHeader`, `TableBody`	Tabele T1, T2, T5
`switch.tsx`	`Switch`	Dark mode toggle
`checkbox.tsx`	`Checkbox`	Filtrowanie wierszy
`button.tsx`	`Button`	Akcje UI
`tooltip.tsx`	`Tooltip`, `TooltipContent`	Podpowiedzi
`badge.tsx`	`Badge`	Oznaczenia
`card.tsx`	`Card`	Karty sekcji
`dropdown-menu.tsx`	`DropdownMenu`	Menu kontekstowe

Utilities

`cn()` — `src/lib/utils.ts`

```
import { cn } from "@/lib/utils"

cn("base-class", condition && "conditional-class", "another-class")
// łączy klasy przez clsx, usuwa konflikty przez tailwind-merge
```

`formatRowIndexId(index)`

Zamienia indeks wiersza na literę Excel: `0→A`, `1→B`, …, `25→Z`, `26→AA`.

Struktura projektu

Struktura projektu

Drzewo katalogów

```
combo-raport-app/
├── src/
│   ├── App.tsx                    # główny komponent aplikacji
│   ├── main.tsx                   # punkt wejścia React (root render)
│   ├── index.css                  # globalne style, zmienne CSS Tailwind
│   ├── components/
│   │   ├── theme-provider.tsx     # kontekst dark/light/system + skrót klawiszowy D
│   │   └── ui/                    # komponenty shadcn/ui
│   │       ├── table.tsx
│   │       ├── checkbox.tsx
│   │       ├── switch.tsx
│   │       ├── button.tsx
│   │       ├── tooltip.tsx
│   │       └── ...
│   └── lib/
│       └── utils.ts               # cn() = clsx + tailwind-merge
├── public/
│   └── data/
│       ├── T1/                    # wolumen miesięczny (ReportRow[])
│       ├── T2/                    # KPI miesięczne (KpiRow[])
│       └── T5/                    # sprzedaż YTD (ReportRow[])
├── docker/
│   └── nginx.conf                 # konfiguracja nginx (tryb prod)
├── Dockerfile                     # multi-stage build
├── docker-compose.yml             # profile dev/prod
├── .dockerignore
├── vite.config.ts                 # Vite + pluginy
├── tsconfig.json                  # TypeScript (references do app i node)
```

```
├── tsconfig.app.json
├── tsconfig.node.json
├── eslint.config.js                # ESLint flat config
├── .prettierrc                     # Prettier
├── components.json                 # konfiguracja shadcn CLI
├── package.json
├── pnpm-lock.yaml
└── pnpm-workspace.yaml             # allowBuilds (esbuild, msw)
```

Stack technologiczny

Warstwa	Biblioteka / Narzędzie	Wersja
UI framework	React	19
Style	Tailwind CSS	4
Komponenty	shadcn/ui, Base UI	—
Tabele	TanStack Table	v8
Ikony	Phosphor Icons, Lucide React	—
Build	Vite	7
Język	TypeScript	5.9
Package manager	pnpm	10
Fonty	JetBrains Mono Variable, Noto Sans Variable	—
Kontener	Docker Desktop, nginx	1.27-alpine

Skrypty pnpm

```
pnpm dev          # serwer developerski Vite (port 5173)
pnpm build        # tsc -b && vite build → dist/
pnpm preview      # nginx-like preview zbudowanego dist/ (port 4173)
pnpm lint         # ESLint na całym projekcie
pnpm format       # Prettier --write na *.ts i *.tsx
pnpm typecheck    # tsc --noEmit (samo sprawdzenie typów, bez buildu)
```

Vite — pluginy

Plugin	Rola
`@vitejs/plugin-react`	JSX transform, Fast Refresh
`@tailwindcss/vite`	Tailwind CSS v4 jako Vite plugin
`vite-tsconfig-paths`	aliasy `@/` z tsconfig
`vite-plugin-svgr`	import SVG jako React komponent
`vite-plugin-checker`	TypeScript + ESLint overlay w trakcie dev
`vite-plugin-live-reload`	przeładowanie na zmianę plików JSON w `public/data/`
`vite-plugin-watch-and-run`	full-reload przez WS po zmianie JSON
`rollup-plugin-visualizer`	generuje `stats.html` z analizą bundle

> `stats.html` generuje się po każdym `pnpm build`. Otwórz go w przeglądarce żeby zobaczyć rozkład rozmiarów modułów.

Aliasy ?cie?ek

```
// tsconfig.app.json + vite.config.ts
"@/*" → "./src/*"
```

Przykład: `import { cn } from "@/lib/utils"`

Konfiguracja shadcn

Plik `components.json` określa:

- styl: `default`
- katalog komponentów: `src/components/ui`
- aliasy CSS: `@/`

Dodawanie nowych komponentów:

```
npx shadcn@latest add <nazwa>
# np. npx shadcn@latest add dialog
```