Specyfikacja aplikacji SizeSync

0) Założenia i cele MVP (skrót)

- Rdzeń: **Zunifikowany Profil Rozmiarowy (ZPR)** z podejściem *measurement-first* (EN 13402), z opcją szybkiego dodania rozmiaru „z metki".
- Funkcje B2C: Zaufany Krąg (Premium 4 osoby, Premium+ bez limitu, dostęp per kategoria), Secret Giver (płatny token; blokada płatna we Free/Premium, darmowa w Premium+), Kalendarz okazji (wewnętrzny), Wishlista (linki).
- Nowe dodatki (włączone do MVP): **mikro-misje**, **powiadomienia kontekstowe**, **automatyczne konwersje między markami**, **Prezentowy link**.
- Monetyzacja (PLN): Premium 10/m, Premium+ 17/m, Secret Giver 20 (token), blokada Secret Giver 5/m (zgodnie z Twoją dyspozycją).

1) Architektura

1.1 Frontend (PWA, offline-first)

- **Next.js (React + TypeScript)**: SSR/SSG dla szybkości i SEO PWA; **React Query** do danych, **Zustand** do lekkiego stanu UI.
- **Tailwind + Headless UI**; i18n: **react-i18next** (PL/EN).
- **Service Worker** + **IndexedDB** (via Dexie): cache profilu, marek, konwersji, tutoriali, tłumaczeń.
- **Tryb offline**: odczyt i edycja ZPR, lokalne kolejkowanie mutacji; funkcje społecznościowe/zakupy działają tylko online (czytelne oznaczenia).

1.2 Backend i infra

- **Supabase** (PostgreSQL EU-region, Auth, Storage, Edge Functions) — szybkie MVP, RLS, gotowe OAuth.
- **Stripe** (web) + **RevenueCat** (agregacja IAP App Store/Google Play po fazie Flutter) do spójnej logiki subskrypcji.
- E-mail: **Postmark/SendGrid**; Push: **Firebase Cloud Messaging (Web Push)**.
- CRONy/planery: **Supabase Scheduled/Edge Functions** do powiadomień kontekstowych, tygodniowych podsumowań i wygasających linków.

1.3 Bezpieczeństwo i prywatność

- **RLS** per wiersz (Postgres) dla tabel profili/udostępnień; wszystkie wrażliwe operacje (Secret Giver, Prezentowy link) tylko via **Edge Functions** z kontrolą uprawnień.
- Szyfrowanie in-transit (TLS) i at-rest; jasne zgody (opt-in), „zero niespodzianek", prawo do bycia zapomnianym (anonimizacja/soft-delete).

2) Model danych (tabele kluczowe)

2.1 Użytkownicy i profil

- users (auth)
- profiles: user_id (PK/FK), display_name, birth_year, sex_optional, unit_pref (metric/imperial), locale, plan (free/premium/premium_plus), settings_json (powiadomienia, prywatność)

2.2 ZPR i historia

- measurements: id, user_id, category (tops/bottoms/shoes/jewelry/… zgodnie z zakresem), **klucze EN 13402** np. chest_cm, waist_cm, hip_cm, foot_len_cm, …, updated_at
- size_labels: szybkie wpisy „z metki": id, user_id, category, brand_id, label_value (np. L/42/US 8), fit_notes, confidence
- measurement_history: wersjonowanie zmian + źródło (ręcznie/system). (Powiadomienia o aktualizacji rozmiarów dzieci).

2.3 Marki i konwersje (nowe: auto-konwersje)

- brands: id, name, country
- brand_size_maps: **rdzeń konwersji** — wiersze odzwierciedlają mapowanie **zakresów pomiarów → etykieta marki**: id, brand_id, category, gender, label_value, min_chest_cm…, max_chest_cm…, min_foot_len_cm…, source (oficjalna tabela), version, valid_from, valid_to
- brand_equivalences: opcjonalne mosty między markami/regionami (np. EU↔US) z confidence_score.
- **Algorytm**: (1) ustal „źródło prawdy" — pomiary ZPR; (2) jeśli użytkownik dodał tylko „metkę", weź brand A + label A → odczytaj odpowiadające zakresy → zmapuj na brand B (i regiony). (Priorytet: oficjalne tabele). **Ten moduł działa w trybie offline dla brandów w cache.**

2.4 Udostępnianie, kręgi, linki

- trust_circles: id, owner_id, name, plan_limit (Premium=4, Premium+=∞)
- trust_circle_members: circle_id, member_user_id, role
- share_rules: poziom **per kategoria**: id, owner_id, target_user_id or circle_id, category, scope (read), expires_atlnull.
- **Prezentowy link** (nowe): shared_links: id, owner_id, scope (np. wishlist_only lub sizes_subset), categories[], token, created_at, expires_at, max_views, views_count, password_hashlnull, one_time (bool).
    - Widok tylko do odczytu, bez logowania; linky domyślnie **wygasają** (np. 7 dni), konfigurowalne przy tworzeniu.

2.5 Secret Giver + blokada

- secret_giver_requests: id, requester_id, target_id, category, status (pending/approved/denied/expired), granted_until, anonymous (bool), rozliczenie tokenu.
- privacy_blocks: owner_id, feature ('secret_giver'), active_until (abonament), polityka planu (Premium+ ma active_until = +∞).

2.6 Misje i powiadomienia

- **Mikro-misje** (nowe): missions: id, code, title, desc, type (once/weekly), reward_type (badge/token/discount), reward_payload, starts_at, ends_at; mission_progress: mission_id, user_id, progress, completed_at.
- **Kontekst** (nowe): notification_rules: id, rule_type (time/event/metric), payload_json (np. „urodziny za 3 tygodnie”, „buty >24m”), channel (push/email), opt_in (bool); notification_log.

3) Logika funkcji (przepływy)

3.1 ZPR: pomiary + „metka”

1. Onboarding ➜ tutorial ➜ ekran ZPR.
2. Użytkownik może: (A) **wprowadzić pomiary wg poradnika** (wizualny, z kontrolą jednostek), lub (B) dodać **etykietę z metki** (brand + label).
3. Dla (B) system natychmiast proponuje **auto-konwersję** do innych marek/regionów (z odznaczeniem poziomu pewności i linkiem do tablicy marki).

3.2 Zaufany Krąg (per kategoria)

- Premium: zaproszenia do 4 osób; Premium+ bez limitu. Właściciel ustawia **share_rules** per kategoria (np. „partner: tops, bottoms, shoes; mama: ring").

3.3 Secret Giver + blokada

- Flow: A (darczyńca) kupuje token 20 PLN ➜ wysyła żądanie kategorii do B ➜ B dostaje jasny ekran zgody ➜ po akceptacji A dostaje **jednorazowy, czasowy** dostęp tylko do wskazanej kategorii. Blokada: Free/Premium 5 PLN/m, Premium+ za darmo w ustawieniach.

3.4 Prezentowy link (nowe)

- Użytkownik generuje **tymczasowy link** (wishlist + niezbędne rozmiary) dla osób bez konta; parametry: czas ważności, jednorazowość, licznik odsłon, opcjonalne hasło. (W Premium+ w przyszłości linki rozszerzone — ale w MVP zaczynamy od wishlist_only + „rozmiary bazowe"). *Pokrewne do „zaawansowanych linków do udostępniania" z planu, adaptowane do B2C.*

3.5 Mikro-misje (nowe)

- Cotygodniowe/okazjonalne: „Uzupełnij 5 kategorii w ZPR", „Dodaj 2 osoby do Kręgu", „Utwórz Prezentowy link przed urodzinami bliskiej osoby".
- Nagrody: badge w profilu, zniżki na 1 miesiąc Premium/Premium+ (kupony Stripe), darmowa blokada Secret Giver na 14 dni.

3.6 Powiadomienia kontekstowe (nowe)

- Reguły: „Buty starsze niż 24 mies.", „urodziny w 21/7/3 dni", „ostatnia aktualizacja wymiarów > 12 mies.", „nowa marka dodana → sprawdź konwersje".
- Kanały per użytkownik (push/e-mail) + „tryb cichy".

3.7 Kalendarz okazji + Weekly Digest

- Wewnętrzny kalendarz: urodziny/rocznice; tygodniowe podsumowanie (ile osób obejrzało Twój profil/wishlistę, nadchodzące okazje, tip tygodnia).

4) API/Edge Functions (wybrane end-pointy)

- POST /api/zpr/measurements (create/update)

- POST /api/size-labels (szybkie dodanie „metki”)
- GET /api/conversions/recommend: **input** {brand_from?, label?, measurements?; brand_to?, category, gender} → **output** {label_suggestion[], confidence} (działa offline z cache; online aktualizuje mapy)
- POST /api/circles/invite, POST /api/shares/set-rule (per kategoria)
- POST /api/secret-giver/request → POST /api/secret-giver/resolve (approve/deny) → GET /api/secret-giver/grant (czasowy token dostępu)
- POST /api/shared-links (tworzenie Prezentowego linku) → zwraca https://sizesync.link/{token}; DELETE /api/shared-links/{id}
- POST /api/missions/progress
- POST /api/notifications/compute (CRON), POST /api/digest/weekly (CRON)

5) UX/UI: nawigacja i stylistyka

5.1 Nawigacja (5 zakładek)

- **Home**: skróty (Dodaj rozmiar / Dodaj metkę / Konwersje), „Co nowego”, karta misji, skrót do Prezentowego linku.
- **Profile**: Twój ZPR (karty: Odzież/Obuwie/Akcesoria/Biżuteria), historia zmian.
- **Krąg**: lista osób, uprawnienia per kategoria, status udostępnień.
- **Gifts**: Wishlista, Secret Giver, Prezentowy link, kalendarz okazji.
- **Ustawienia**: plan, płatności, powiadomienia, język/jednostki, prywatność.

5.2 Paleta i tryby

- **Light**: tło #F7F7F7, tekst #1F2937, **Primary** #4C5B5C, **Accent** (CTA) #E07A5F, **Success** #81B29A.
- **Dark**: tło #111315, tekst #E7E7E7, **Primary** #93A1A1 (odcienie tealu w dark), **Accent** i **Success** jak wyżej.
- Auto-detect systemowego motywu + przełącznik w Ustawieniach.
- Typografia: humanistyczny sans (Inter/DM Sans), duże kontrasty, sporo „powietrza”, brak pstrokacizny.

5.3 Ekrany kluczowe

- **Dodaj metkę**: 3 kroki (marka → kategoria → etykieta), wynik: auto-konwersje (lista marek z „pewnością”).
- **Prezentowy link**: zakres, TTL, jednorazowość, hasło (opcjonalne), przycisk „Kopiuj”; ekran podglądu linku gościa (wishlist + wymagane rozmiary).
- **Misje**: lista celów, pasek postępu, odbiór nagrody.
- **Zgoda Secret Giver**: prosty, krótki, jasny język; „kto/co/dokładnie na jak długo”.

6) Analityka, KPI, retencja

6.1 KPI (MVP)

- **Akwizycja**: rejestracje, aktywacje ZPR (≥1 kategoria), „Dodaj metkę” użyte ≥1 raz.
- **Zaangażowanie**: DAU/WAU/MAU, ukończone **misje**, liczba **Prezentowych linków**, liczba **konwersji między markami**.
- **Social**: liczba członków Kręgu na użytkownika, odsetek z rule-sharing per kategoria.
- **Monetyzacja**: konwersja Free→Premium/Premium+, ARPPU, wykorzystania Secret Giver i blokad.
- **Retencja**: D1/D7/D30, procent użytkowników z włączonymi powiadomieniami kontekstowymi.

6.2 Telemetria (RODO-friendly)

- **Plausible/Matomo** + eventy własne (self-hosted); odrębny strumień backend (agregowane) dla misji/powiadomień; brak PII bez zgody.

7) Plan wdrożenia i harmonogram (z boosterami)

Faza	Czas	Zakres (skrót)
0. Projekt	2 tyg.	UX flowy, design system, i18n, słowniki kategorii
1. Rdzeń	4 tyg.	Auth, ZPR (pomiary + metka), historia, marki + mapy, auto-konwersje v1
2. Social	2 tyg.	Krąg (limity planów), udostępnianie per kategoria, Prezentowy link v1
3. Monetyzacja	2 tyg.	Stripe + plany + Secret Giver + blokada, ustawienia płatności

4. Retencja	2 tyg.	Powiadomienia kontekstowe (CRON), mikro-misje, tygodniowy digest
5. Polerka/QA	2 tyg.	Testy, dostępność, performance, hardening RLS, DPIA/RODO

Szacunkowo: ~12–14 tygodni (2–3 osoby dev + 0.5 UX).
(Booster-funkcje zostały rozłożone tak, by nie „rozerwać" MVP.)

8) Testy i jakość

- **Jednostkowe/integracyjne** (Vitest/RTL); e2e: Playwright (scenariusze: Secret Giver, Prezentowy link, offline edycja i sync).
- **Testy dostępności** (axe), kontrasty, nawigacja klawiaturą.
- **Testy offline**: throttle sieci, konflikty przy synchronizacji (strategia *last-write-wins* + dziennik zmian).
- **Bezpieczeństwo**: audyt RLS (kto czyta co), testy Edge Functions, rate-limity dla linków/SG.

9) Cennik i logika planów (MVP) – zaktualizowane

Zasada: funkcje są **narastające** – Premium zawiera wszystko z Free + dodatkowe, Premium+ zawiera wszystko z Premium + dodatkowe.

Plan	Cena (PLN/m)	Zaufany Krąg	Blokada SG	Inne kluczowe funkcje
Free	0	1 osoba	Płatna 5 PLN/m	Pełny ZPR (pomiary + metka), auto-konwersje między markami, Prezentowy link (wishlist + bazowe rozmiary, TTL 7 dni, max_views=5), Kalendarz okazji, Powiadomienia kontekstowe, Mikro-misje
Premium	10	4 osoby	Płatna 5 PLN/m	Wszystko z Free + rozszerzone udostępnianie (per kategoria), historia zmian wymiarów z sugestiami, bardziej rozbudowany Kalendarz (więcej typów okazji), dodatkowe typy mikro-misji, raport tygodniowy z większą ilością szczegółów
Premium+	17	Bez limitu	W cenie	Wszystko z Premium + Prezentowy link w wersji zaawansowanej (więcej kategorii, dłuższy TTL, brak limitu odsłon, hasło), dodatkowe typy powiadomień kontekstowych, priorytetowe wsparcie

10) Ekrany ustawień i zgody (UE)

- **Prywatność**: przejrzyste przełączniki (SG: włącz/wyłącz; Prezentowe linki: lista aktywnych + REVOKE).

- **Zgody**: powiadomienia push/e-mail granularnie; telemetry opt-in; wiek 16+ lub zgoda rodzica.
- **DPIA**: rejestr czynności, minimalizacja danych, *purpose limitation*, retention (np. auto-usuwanie logów powiadomień po 18 mies.).

11) Ryzyka i jak je neutralizujemy

- **Mapy marek**: oficjalne tabele bywają niepełne → wersjonowanie i źródło, „confidence” + komunikaty („przybliżone”). Cache w PWA dla offline.
- **Sekret/Linki**: wycieki przez zbyt długi TTL → domyślnie krótkie wygasanie (7 dni), max_views, opcjonalne hasło, możliwość natychmiastowego „REVOKE”.
- **Retencja**: niska aktywność po wprowadzeniu danych → **misje**, **kontekstowe powiadomienia**, **digest tygodniowy** od dnia 1.

12) Co przygotuję do startu dev (artefakty)

- Specyfikacja OpenAPI (end-pointy powyżej) + schemat DB (ERD).
- Biblioteka komponentów (Tailwind), motywy light/dark.
- Zasady RLS (SQL) i Edge Functions (TS) dla: Secret Giver, Prezentowy link, rozliczenia planów.
- Pakiet testów e2e (szkielety) i checklista dostępności.

Prezentowy link — domyślne parametry (MVP)

Zakres danych w MVP: wishlist + bazowe rozmiary (kategorie wskazane przez użytkownika).
Bez konta po stronie odbiorcy. Link zawsze można ręcznie „Revoke”.

Parametr	Free	Premium	Premium+
Domyślny TTL (ważność)	7 dni	14 dni	30 dni
Zakres kategorii	1–3 wybrane	do 6	dowolne
Limit odsłon (max_views)	5	20	brak limitu (można ustawić własny)
Jednorazowy dostęp	✔	✔	✔
Hasło do linku	—	—	✔ (opcjonalne)

Data/godzina wygaśnięcia	automatyczna (23:59 dnia TTL)	j.w.	j.w. lub ręczna
Powiadomienie o zbliżającym się wygaśnięciu	24 h przed	48 h i 24 h	72 h i 24 h
Log dostępu (anonimizowany)	✔ (licznik)	✔ (licznik + stemple czasu)	✔ (licznik + stemple czasu)

Dodatki techniczne:

- Domyślne ustawienia można nadpisać przy tworzeniu linku (w ramach limitów planu).
- Token linku: losowy, jednorazowo generowany; opcjonalnie **one_time=true** (pierwszy odczyt wygasza).
- Retencja logu: 90 dni (anonimizowane), potem agregat statystyczny.

Powiadomienia kontekstowe — progi i domyślne reguły

Kanały: push / e-mail (użytkownik wybiera; domyślnie push ON, mail ON).
Godziny ciszy: domyślnie 22:00–8:00 (lokalnie), konfigurowalne.
Częstotliwość CRON:

- dzienny: 07:30 (digest okoliczności na 24–72 h),
- tygodniowy: poniedziałek 08:00 (mini-podsumowanie),
- natychmiastowe: zdarzenia (np. prośba Secret Giver, limit linku).

Obszar	Reguła (warunek)	Kiedy wysyłamy	Notatki
Zbliżając e się	urodziny/rocznice z kalendarza	21, 7, 3, 1 dzień przed (push), + przypomnienie w	Eskalacja: jeśli brak Prezentowego linku 7 dni przed — sugestia
Starość rozmiaru (dorośli)	ostatnia aktualizacja danej kategorii > 12 mies.	raz / 12 mies. od ostatniej zmiany	Dla butów: próg 24 mies. (lub 18–24 z losowym oknem, by uniknąć „efektu masowego pingowania”)
Dzieci —	profil z oznaczeniem „dziecko”	co 3 mies. przypomnienie aktualizacji stopy/odzieży	Komunikat przyjazny, bez presji
Braki w	< 3 kategorie	raz/tydz. przez 4 tyg.,	Proponujemy mikro-misję „Uzupełnij
Aktywno ść Kręgu	nowe wyświetlenia profilu lub wishlisty	dzienny zbiorczy push (max 1/dzień)	Tylko licznik/ogólny wgląd, bez PII
Prezento wy link	1) zbliża się wygaśnięcie, 2)	24/48/72 h przed; natychmiast po osiągnięciu	Z przyciskiem „Przedłuż” / „Resetuj limit” (wg planu)
Secret Giver —	nowa prośba / wygasanie dostępu	natychmiast / 24 h przed wygaśnięciem	Treść „kto/co/jak długo” i szybkie przyciski: Akceptuj/Odrzuć
Misje	cel bliski	max 1/dzień	Np. „Brakuje 1 kroku do odznaki”

Tygodniowy	aktywność i „co dalej”	poniedziałek 08:00	Krótko i wartościowo (patrz niżej)

Tygodniowy digest (MVP):

- „X osób obejrzało Twoją wishlistę/rozmiary w tym tygodniu” (tylko liczba).
- „Najbliższe okazje: …” (7–21 dni).
- „Sugestie kolejnych kroków”: dokończ ZPR / zaproś do Kręgu / utwórz Prezentowy link.

Mikro-misje / wyzwania — pakiet startowy

Zasada: krótkie, osiągalne, widoczna nagroda. Nagrody: odznaki, darmowe dni blokady SG, kupony na –20% (1 miesiąc Premium/Premium+) lub 7-dniowe triale. Misje resetują się co tydzień (typu weekly) lub są jednorazowe (once). Wszystko w pełni **opcjonalne**.

Kod	Typ	Treść misji	Warunek	Nagroda (przykłady)
ONBOARD_1	once	Uzupełnij przynajmniej 3 kategorie z ZPR	≥3 kategorie z [illegible]	Odznaka „Start” + 7 dni [illegible]
ONBOARD_2	once	Dodaj 1 rozmiar „z metki”	`size_label`	–20% na 1. miesiąc Premium [illegible]
WEEKLY_ZPR	weekly	Zaktualizuj dowolny wymiar	dowolna edycja w `measurements`	1 token misji (zbieraj 4 → 1 tydzień Premium)
WEEKLY_CONV	weekly	Sprawdź konwersje w 2 różnych markach	≥2 zapytania do konwersji	Odznaka „Znawca marek”
WEEKLY_LINK	weekly	Utwórz Prezentowy link przed nadchodzącą okazją	link z TTL ≤14 dni, ≥1 view	+7 dni blokady SG (stackowalne do 28 dni)
CIRCLE_1	once	Zaproś 1 osobę do Zaufanego Kręgu	`trust_circle_member` added	Odznaka „Mój Krąg”
CIRCLE_2	weekly	Ustaw zasady udostępnień per kategoria	≥1 `share_rule` zmodyfikowana	5 dodatkowych odsłon do najbliższego Prezentowego
CARE_KIDS	quarterly	Zaktualizuj rozmiary dziecka	zmiana w profilu z tagiem child	Odznaka „Opiekun” + przypinka w profilu
PRIVACY_OK	once	Skonfiguruj prywatność i [illegible]	zapis w [illegible]	Odznaka „Świadomy”

Uwagi implementacyjne:

- Misje i nagrody zapisujemy w missions / mission_progress; nagrody monetarne rozliczamy kuponami Stripe (MVP).
- Tokeny/bonusy „blokady SG” aplikujemy jako wpisy w privacy_blocks z active_until (+X dni).
- Każdą misję da się „odrzucić” (nie przypominamy o niej przez 60 dni).

Drobne doprecyzowania (dla dev i QA)

- **Copy UX (PL/EN)**: wszystkie powiadomienia mają krótkie, nieinwazyjne teksty i zawsze *jedną* rekomendowaną akcję.
- **Opt-in per kategoria powiadomień** (Okazje / ZPR / Krąg / Linki / SG / Misje / Digest).
- **A/B ready**: progi „12 vs 9 mies." dla dorosłych i „3 vs 4 mies." dla dzieci — parametry konfigurowalne z panelu admin (feature flags), bez deployu.
- **Rate-limit**: max 1 powiadomienie kontekstowe/6 h (z wyjątkiem natychmiastowych akcji, np. prośba SG).
- **Dostępność**: wszystkie odznaki i komunikaty mają opisy ARIA; linki „Revoke/Extend" dostępne z klawiatury.

Moduł 1 – User Stories (MVP)

1. Autoryzacja i profil użytkownika

US-1 – Rejestracja/logowanie

- **Opis:** Jako nowy użytkownik chcę założyć konto przy użyciu e-maila i hasła lub konta Google/Apple, aby szybko rozpocząć korzystanie z aplikacji.
- **AC:**
    1. Mogę wybrać metodę rejestracji: e-mail/hasło lub OAuth.
    2. Po rejestracji trafiam na ekran samouczka.
    3. Walidacja formularza: e-mail poprawny, hasło min. 8 znaków.
    4. Zapis profilu w bazie (ustawienia domyślne).

US-2 – Edycja ustawień profilu

- **Opis:** Jako użytkownik chcę edytować swoje dane (język, jednostki, powiadomienia, plan), aby dostosować aplikację do swoich potrzeb.
- **AC:**
    1. Mogę zmienić język (PL/EN).
    2. Mogę ustawić jednostki (metryczne/dom).
    3. Mogę włączyć/wyłączyć powiadomienia push/e-mail per kategoria.
    4. Zmiany zapisują się w bazie i są widoczne po ponownym uruchomieniu aplikacji.

2. Zunifikowany Profil Rozmiarowy (ZPR)

US-3 – Dodanie wymiarów ciała (*measurement-first*)

- **Opis:** Jako użytkownik chcę dodać swoje dokładne wymiary ciała według standardu EN 13402, aby aplikacja mogła sugerować odpowiednie rozmiary.
- **AC:**
    1. Formularz z wizualnymi instrukcjami (gdzie mierzyć).
    2. Mogę podać wartości w cm lub calach (konwersja automatyczna).
    3. Dane zapisywane w tabeli measurements.
    4. Historia zmian tworzona przy każdej edycji.

US-4 – Dodanie rozmiaru z metki

- **Opis:** Jako użytkownik chcę szybko zapisać rozmiar z metki ubrania, aby móc uzyskać jego odpowiedniki w innych markach.
- **AC:**
    1. Formularz: wybór marki → kategoria → rozmiar z listy.
    2. System pokazuje konwersje do innych marek (confidence score).
    3. Dane zapisywane w tabeli size_labels.

US-5 – Auto-konwersje między markami

- **Opis:** Jako użytkownik, który dodał rozmiar z metki, chcę zobaczyć automatyczne konwersje tego rozmiaru na inne marki, aby łatwiej robić zakupy.
- **AC:**
    1. Algorytm działa offline dla brandów w cache.
    2. Wskazuje poziom pewności (np. 95% – oficjalna tabela).
    3. Link do tabeli rozmiarów danej marki w widoku szczegółowym.

3. Udostępnianie i Zaufany Krąg

US-6 – Tworzenie Zaufanego Kręgu

- **Opis:** Jako użytkownik chcę dodać osoby do mojego Kręgu, aby mogły zobaczyć moje rozmiary w wybranych kategoriach.
- **AC:**

1. Limit osób wg planu (Free=1, Premium=4, Premium+=∞).
2. Mogę ustawić uprawnienia per kategoria.
3. Zaproszenie wysyłane e-mailem lub kodem.

4. Secret Giver

US-7 – Wysyłanie prośby SG

- **Opis:** Jako darczyńca chcę wysłać prośbę o dostęp do wybranej kategorii rozmiarów, aby kupić prezent.
- **AC:**
  1. Muszę mieć aktywny token SG (kupiony — koszt 20 PLN, dotyczy wszystkich planów).
  2. Wybieram kategorię i czas dostępu (np. 48 h).
  3. Odbiorca dostaje powiadomienie push/e-mail.
  4. Po akceptacji mam dostęp tylko do wybranej kategorii, przez czas określony w prośbie.
  5. Mogę wysłać kolejną prośbę tylko po zakupie nowego tokenu.

5. Prezentowy link

US-8 – Tworzenie Prezentowego linku

- **Opis:** Jako użytkownik chcę wygenerować link z wishlistą i wybranymi rozmiarami, aby osoba bez konta mogła zobaczyć te dane.
- **AC:**
  1. Mogę ustawić TTL i kategorie wg limitów planu.
  2. Free: TTL=7 dni, max_views=5; Premium: 14 dni, 20 odsłon; Premium+: 30 dni, bez limitu.
  3. Mogę natychmiast unieważnić link.

6. Mikro-misje

US-9 – Ukończenie misji

- **Opis:** Jako użytkownik chcę wykonywać misje, aby otrzymywać odznaki i nagrody.
- **AC:**
  1. Misje weekly i once są widoczne w zakładce „Misje”.

2. System automatycznie zalicza postęp po wykonaniu akcji.
3. Po ukończeniu mogę odebrać nagrodę (np. kupon, token).

7. Powiadomienia kontekstowe

US-10 – Otrzymywanie przypomnień

- **Opis:** Jako użytkownik chcę otrzymywać powiadomienia, gdy zbliżają się okazje lub kiedy powinienem zaktualizować rozmiary.
- **AC:**
  1. Mogę włączyć/wyłączyć powiadomienia per kategoria.
  2. Powiadomienia wysyłane zgodnie z ustalonymi progami.
  3. Mogę ustawić godziny ciszy.

2.1. Typy i słowniki

sql
KopiujEdytuj

```
-- Kategorie główne (rozszerzalne)
create type category as enum
('tops','bottoms','shoes','outerwear','underwear','accessories','jewelry','other');

-- Płeć/target (do map tablic rozmiarów – nie tożsamość)
create type target_gender as enum ('women','men','unisex','kids');

-- Jednostki preferowane przez użytkownika
create type unit_pref as enum ('metric','imperial');

-- Plany
create type user_plan as enum ('free','premium','premium_plus');

-- Status prośby SG
create type sg_status as enum ('pending','approved','denied','expired','revoked');

-- Scope udostępnienia (per kategoria – read-only)
create type share_scope as enum ('read');

-- Scope prezentowego linku (MVP)
create type shared_link_scope as enum ('wishlist_only','wishlist_plus_sizes');

-- Kanały powiadomień
create type notif_channel as enum ('push','email');
```

```
-- Typy misji
create type mission_type as enum ('once','weekly','quarterly');

-- Rodzaj nagrody
create type reward_type as enum ('badge','sg_block_days','coupon','token');

-- Źródło mapy rozmiarów
create type size_source as enum ('official','derived','community','import');
```

2.2. Użytkownicy i profil

sql
KopiujEdytuj

```
-- Profil użytkownika (1:1 z auth.users)
create table profiles (
  user_id uuid primary key references auth.users(id) on delete cascade,
  display_name text,
  birth_year int,
  is_child boolean default false,
  unit_preference unit_pref not null default 'metric',
  locale text not null default 'pl-PL',
  plan user_plan not null default 'free',
  settings_json jsonb not null default '{}'::jsonb, -- powiadomienia, prywatność,
godziny ciszy
  created_at timestamptz not null default now(),
  updated_at timestamptz not null default now()
);

create index idx_profiles_plan on profiles(plan);
```

2.3. ZPR – wymiary, metki, historia

sql
KopiujEdytuj

```
-- Wymiary wg EN 13402 (przykładowe pola; dokładny zestaw możemy rozszerzyć)
create table measurements (
  id uuid primary key default gen_random_uuid(),
  user_id uuid not null references profiles(user_id) on delete cascade,
  category category not null,
  -- górne partie
  chest_cm numeric(6,2),
  waist_cm numeric(6,2),
  hip_cm numeric(6,2),
```

```
  -- dolne partie
  inseam_cm numeric(6,2),
  outseam_cm numeric(6,2),
  -- stopy
  foot_len_cm numeric(6,2),
  -- biżuteria
  ring_inner_diam_mm numeric(6,2),
  wrist_circ_mm numeric(6,2),
  neck_circ_cm numeric(6,2),
  notes text,
  updated_at timestamptz not null default now(),
  unique (user_id, category)
);

create index idx_measurements_user on measurements(user_id);

-- Historia zmian (wersjonowanie)
create table measurement_history (
  id uuid primary key default gen_random_uuid(),
  measurement_id uuid not null references measurements(id) on delete cascade,
  snapshot jsonb not null,               -- pełny poprzedni stan rekordu measurements
  source text not null,                  -- 'user','system','import'
  created_at timestamptz not null default now()
);
create index idx_mhist_mid on measurement_history(measurement_id);

-- Szybkie wpisy "z metki"
create table size_labels (
  id uuid primary key default gen_random_uuid(),
  user_id uuid not null references profiles(user_id) on delete cascade,
  brand_id uuid,                         -- opcjonalnie null, gdy marka spoza bazy
  brand_name text,                       -- redundancja dla UX i gdy brand_id null
  category category not null,
  target target_gender default 'unisex',
  label_value text not null,             -- np. 'M', '42', 'US 8'
  region text,                           -- np. 'EU','US','UK'
  fit_notes text,
  confidence numeric(5,2),               -- 0-100; jeżeli po auto-mapowaniu
  created_at timestamptz not null default now()
);
create index idx_sizelabels_user on size_labels(user_id);
create index idx_sizelabels_brand on size_labels(brand_id);
```

2.4. Marki i mapy rozmiarów (auto-konwersje)

sql
KopiujEdytuj

```
create table brands (
  id uuid primary key default gen_random_uuid(),
  name text not null unique,
  country text,
  website text,
  created_at timestamptz not null default now()
);

-- Mapowanie zakresów pomiarów -> etykieta rozmiaru dla konkretnej marki i kategorii
create table brand_size_maps (
  id uuid primary key default gen_random_uuid(),
  brand_id uuid not null references brands(id) on delete cascade,
  category category not null,
  target target_gender not null default 'unisex',
  region text not null default 'EU',
  label_value text not null,              -- np. 'M', '42', '8'
  -- Zakresy pomiarów (wg kategorii – puste pola dozwolone)
  min_chest_cm numeric(6,2),
  max_chest_cm numeric(6,2),
  min_waist_cm numeric(6,2),
  max_waist_cm numeric(6,2),
  min_hip_cm numeric(6,2),
  max_hip_cm numeric(6,2),
  min_inseam_cm numeric(6,2),
  max_inseam_cm numeric(6,2),
  min_foot_len_cm numeric(6,2),
  max_foot_len_cm numeric(6,2),
  min_ring_inner_diam_mm numeric(6,2),
  max_ring_inner_diam_mm numeric(6,2),
  source size_source not null default 'official',
  version int not null default 1,
  valid_from date default current_date,
  valid_to date,
  created_at timestamptz not null default now()
);

create index idx_bsm_brand_cat on brand_size_maps(brand_id, category);
create index idx_bsm_label on brand_size_maps(label_value);
```

```
-- (opcjonalnie) powiązania między markami/regionami
create table brand_equivalences (
  id uuid primary key default gen_random_uuid(),
  from_brand_id uuid not null references brands(id) on delete cascade,
  to_brand_id uuid not null references brands(id) on delete cascade,
  category category not null,
  target target_gender not null default 'unisex',
  region_from text not null default 'EU',
  region_to text not null default 'US',
  confidence numeric(5,2) not null default 80,
  created_at timestamptz not null default now(),
  unique (from_brand_id, to_brand_id, category, target, region_from, region_to)
);
```

2.5. Zaufany Krąg i reguły udostępniania

sql
KopiujEdytuj

```
create table trust_circles (
  id uuid primary key default gen_random_uuid(),
  owner_id uuid not null references profiles(user_id) on delete cascade,
  name text not null,
  created_at timestamptz not null default now()
);
create index idx_circles_owner on trust_circles(owner_id);

create table trust_circle_members (
  circle_id uuid not null references trust_circles(id) on delete cascade,
  member_user_id uuid not null references profiles(user_id) on delete cascade,
  role text not null default 'member',
  added_at timestamptz not null default now(),
  primary key (circle_id, member_user_id)
);

-- Reguły: per kategoria, read-only
create table share_rules (
  id uuid primary key default gen_random_uuid(),
  owner_id uuid not null references profiles(user_id) on delete cascade,
  target_user_id uuid references profiles(user_id) on delete cascade,
  circle_id uuid references trust_circles(id) on delete cascade,
  category category not null,
  scope share_scope not null default 'read',
  expires_at timestamptz,
  created_at timestamptz not null default now(),
```

```
  check ((target_user_id is not null) or (circle_id is not null))
);
create index idx_share_owner on share_rules(owner_id);
create index idx_share_target on share_rules(target_user_id);
create index idx_share_circle on share_rules(circle_id);
```

2.6. Prezentowy link

sql
KopiujEdytuj

```
create table shared_links (
  id uuid primary key default gen_random_uuid(),
  owner_id uuid not null references profiles(user_id) on delete cascade,
  scope shared_link_scope not null default 'wishlist_only',
  categories category[] not null default '{}',
  token text not null unique,              -- publiczny identyfikator linku
  password_hash text,                      -- null = bez hasła (Premium+ może ustawić)
  one_time boolean not null default false, -- po pierwszym view wygasa
  max_views int,                           -- null = bez limitu (Premium+)
  views_count int not null default 0,
  expires_at timestamptz not null,
  created_at timestamptz not null default now(),
  revoked_at timestamptz
);

create index idx_shared_links_owner on shared_links(owner_id);
create index idx_shared_links_expires on shared_links(expires_at);
```

2.7. Secret Giver (token płatny) + blokada

sql
KopiujEdytuj

```
-- Prośby o dostęp (token zawsze płatny – poza DB rozlicza to Stripe)
create table secret_giver_requests (
  id uuid primary key default gen_random_uuid(),
  requester_id uuid not null references profiles(user_id) on delete cascade,
  target_id uuid not null references profiles(user_id) on delete cascade,
  category category not null,
  status sg_status not null default 'pending',
  requested_hours int not null default 48,
  approved_until timestamptz,              -- ustawiane przy 'approved'
  created_at timestamptz not null default now(),
  updated_at timestamptz not null default now(),
  constraint sg_self check (requester_id <> target_id)
```

```
);
create index idx_sg_target on secret_giver_requests(target_id);
create index idx_sg_requester on secret_giver_requests(requester_id);

-- Blokady funkcji (np. SG) – Premium+ ma „w cenie”
create table privacy_blocks (
  id uuid primary key default gen_random_uuid(),
  owner_id uuid not null references profiles(user_id) on delete cascade,
  feature text not null,                  -- 'secret_giver'
  active_until timestamptz,               -- null = włączone bezterminowo (Premium+)
  created_at timestamptz not null default now(),
  unique (owner_id, feature)
);
```

2.8. Kalendarz okazji, wishlisty

sql
KopiujEdytuj

```
-- Okazje (wewnętrzny kalendarz)
create table events (
  id uuid primary key default gen_random_uuid(),
  user_id uuid not null references profiles(user_id) on delete cascade,
  title text not null,                    -- np. "Urodziny Ani"
  event_date date not null,
  repeat_yearly boolean not null default true,
  notes text,
  created_at timestamptz not null default now()
);
create index idx_events_user on events(user_id);
create index idx_events_date on events(event_date);

-- Wishlista
create table wishlists (
  id uuid primary key default gen_random_uuid(),
  user_id uuid not null references profiles(user_id) on delete cascade,
  name text not null default 'Lista życzeń',
  created_at timestamptz not null default now()
);
create index idx_wishlist_user on wishlists(user_id);

create table wishlist_items (
  id uuid primary key default gen_random_uuid(),
  wishlist_id uuid not null references wishlists(id) on delete cascade,
  title text not null,
  url text,
```

```
  category category,
  notes text,
  created_at timestamptz not null default now()
);
create index idx_wishlist_items_list on wishlist_items(wishlist_id);
```

2.9. Misje i nagrody

sql
KopiujEdytuj

```
create table missions (
  id uuid primary key default gen_random_uuid(),
  code text not null unique,              -- np. 'ONBOARD_1'
  title text not null,
  description text,
  type mission_type not null,
  reward reward_type not null,
  reward_payload jsonb not null default '{}'::jsonb, -- np. {"days":7} dla
sg_block_days, {"percent":20} dla kuponu
  starts_at timestamptz,
  ends_at timestamptz,
  created_at timestamptz not null default now()
);

create table mission_progress (
  mission_id uuid not null references missions(id) on delete cascade,
  user_id uuid not null references profiles(user_id) on delete cascade,
  progress int not null default 0,
  completed_at timestamptz,
  reward_claimed_at timestamptz,
  primary key (mission_id, user_id)
);
create index idx_mprogress_user on mission_progress(user_id);
```

2.10. Powiadomienia (reguły, log)

sql
KopiujEdytuj

```
-- Reguły (wygenerowane z ustawień użytkownika i defaultów)
create table notification_rules (
  id uuid primary key default gen_random_uuid(),
  user_id uuid not null references profiles(user_id) on delete cascade,
  rule_type text not null,                --
'occasion','stale_size','kids_growth','link_expiry','sg_request','mission','digest'
```

```
  params jsonb not null default '{}'::jsonb,   -- np. { "days":[21,7,3,1] }
  channel notif_channel not null default 'push',
  opt_in boolean not null default true,
  created_at timestamptz not null default now()
);
create index idx_notif_rules_user on notification_rules(user_id);

-- Log wysyłek (agregowany – bez PII w treści)
create table notification_log (
  id uuid primary key default gen_random_uuid(),
  user_id uuid not null references profiles(user_id) on delete cascade,
  rule_id uuid references notification_rules(id) on delete set null,
  kind text not null,                    -- 'push'/'email'
  payload jsonb not null,                -- tytuł, skrót, identyfikatory domenowe
  sent_at timestamptz not null default now()
);
create index idx_notif_log_user on notification_log(user_id);
create index idx_notif_log_sent on notification_log(sent_at);
```

2.11. Subskrypcje i płatności (mirror Stripe/IAP)

Logika rozliczeń jest w Stripe/RevenueCat, ale trzymamy lokalne, odporne na opóźnienia lustro stanu.

sql
KopiujEdytuj

```
create table subscriptions (
  id uuid primary key default gen_random_uuid(),
  user_id uuid not null references profiles(user_id) on delete cascade,
  provider text not null,                -- 'stripe','apple','google'
  external_id text not null,             -- id subskrypcji po stronie providera
  plan user_plan not null,               -- plan docelowy
  status text not null,                  --
'active','incomplete','past_due','canceled','paused'
  current_period_end timestamptz,
  created_at timestamptz not null default now(),
  unique (provider, external_id)
);
create index idx_subs_user on subscriptions(user_id);
```

2.12. Indeksy, integralność, notatki RLS

- Wszystkie FK mają on delete cascade, by zapobiec „osieroconym” rekordom.

- Indeksy do najczęstszych zapytań: brand_size_maps(brand_id, category), size_labels(user_id), shared_links(expires_at), secret_giver_requests(target_id).
- **RLS (do wdrożenia w Moduł 3):**
  - profiles: user_id = auth.uid() (select/update własnego profilu).
  - measurements / size_labels: jak wyżej + czytanie przez uprawnionych (share_rules / SG / shared_links).
  - trust_* / share_rules: właściciel + adresaci.
  - shared_links: tylko właściciel; odczyt przez token via Edge Function.
  - secret_giver_requests: requester/target.
  - missions (global read), mission_progress (właściciel).
  - notification_*: właściciel.

2.13. Minimalne dane startowe (seed)

sql
KopiujEdytuj

```
-- Przykładowe marki
insert into brands (id, name, country) values
  (gen_random_uuid(), 'Acme Apparel', 'PL'),
  (gen_random_uuid(), 'North River', 'US');

-- Przykładowe mapy dla butów (EU)
-- (W realu import oficjalnych tabel od producentów)
-- wartości przykładowe:
-- EU 42: foot_len 26.0–26.6 cm
```

(Rzeczywiste tabele marek załadujemy importem – narzędzia: CSV → brand_size_maps z walidacją źródła i wersji.)

ERD – opis relacji (tekstowo)

- profiles (1) —— (N) measurements, profiles (1) —— (N) size_labels
- measurements (1) —— (N) measurement_history
- brands (1) —— (N) brand_size_maps, brands (N) —— (N) brands przez brand_equivalences

- profiles (1) —— (N) trust_circles —— (N) trust_circle_members (N) —— (1) profiles
- profiles (1) —— (N) share_rules —— (1) profiles *lub* share_rules —— (1) trust_circles
- profiles (1) —— (N) shared_links
- profiles (1) —— (N) secret_giver_requests (jako requester i/lub target)
- profiles (1) —— (N) privacy_blocks
- profiles (1) —— (N) events, profiles (1) —— (N) wishlists —— (N) wishlist_items
- missions (1) —— (N) mission_progress —— (1) profiles
- profiles (1) —— (N) notification_rules —— (N) notification_log

A) RLS — zasady bezpieczeństwa (PostgreSQL / Supabase)

Założenia:
• auth.uid() – identyfikator zalogowanego użytkownika (Supabase).
• **Włączamy RLS** na wszystkich tabelach domenowych.
• Dostęp publiczny (bez logowania) do Prezentowego linku realizujemy **wyłącznie przez Edge Function** z kluczem serwisowym (service role) – RLS pozostaje restrykcyjny.
• Dostępy „pośrednie” (Krąg, reguły udostępniania, SG) egzekwujemy w RLS przez EXISTS (…) na powiązanych tabelach.

A.1. Włączenie RLS

sql
KopiujEdytuj
```
-- Przykład: włączamy RLS na wszystkich tabelach (uruchomić raz)
alter table profiles enable row level security;
alter table measurements enable row level security;
alter table measurement_history enable row level security;
alter table size_labels enable row level security;
alter table brands enable row level security;
alter table brand_size_maps enable row level security;
alter table trust_circles enable row level security;
alter table trust_circle_members enable row level security;
alter table share_rules enable row level security;
alter table shared_links enable row level security;
```

```
alter table secret_giver_requests enable row level security;
alter table privacy_blocks enable row level security;
alter table events enable row level security;
alter table wishlists enable row level security;
alter table wishlist_items enable row level security;
alter table missions enable row level security;
alter table mission_progress enable row level security;
alter table notification_rules enable row level security;
alter table notification_log enable row level security;
alter table subscriptions enable row level security;
```

A.2. Polityki — profile i własne dane

sql
KopiujEdytuj

```
-- PROFILES: właściciel widzi/edytuje własny profil
create policy prof_self_select on profiles
for select using (user_id = auth.uid());

create policy prof_self_update on profiles
for update using (user_id = auth.uid());

-- Brak INSERT/DELETE przez klienta (robimy to Edge/DB triggerami).
```

A.3. ZPR (measurements, size_labels) — właściciel + udostępnienia

Dostęp ma: (1) właściciel, (2) adresat **share_rules** (per kategoria, user lub circle), (3) w czasie aktywnego **Secret Giver** (per kategoria).

Wspólne predykaty:

sql
KopiujEdytuj

```
-- funkcja pomocnicza (opcjonalnie) skracająca zapytania w RLS:
-- create function can_read_category(target_owner uuid, cat category) returns
boolean as $$
--   select
--     target_owner = auth.uid()
--     or exists (
--       select 1 from share_rules sr
--       where sr.owner_id = target_owner
--         and sr.category = cat
--         and sr.scope = 'read'
--         and sr.expires_at is null or sr.expires_at > now()
--         and (sr.target_user_id = auth.uid()
--              or exists (
```

```
--                 select 1 from trust_circle_members tcm
--                 join trust_circles tc on tc.id = tcm.circle_id
--                 where tc.owner_id = target_owner and tcm.member_user_id =
auth.uid()
--             )
--         )
--     )
--     or exists (
--       select 1 from secret_giver_requests sg
--       where sg.target_id = target_owner
--         and sg.status = 'approved'
--         and sg.approved_until > now()
--         and sg.category = cat
--         and sg.requester_id = auth.uid()
--     );
-- $$ language sql stable;
```

measurements

sql

KopiujEdytuj

```
create policy meas_owner_rw on measurements
for all using (user_id = auth.uid())
with check (user_id = auth.uid());

-- Odczyt przez udostępnienia/SG
create policy meas_shared_r on measurements
for select using (
  user_id = auth.uid()
  or exists (
    select 1 from share_rules sr
    where sr.owner_id = measurements.user_id
      and sr.category = measurements.category
      and sr.scope = 'read'
      and (sr.expires_at is null or sr.expires_at > now())
      and (
        sr.target_user_id = auth.uid()
        or exists (
          select 1 from trust_circle_members tcm
          join trust_circles tc on tc.id = tcm.circle_id
          where tc.owner_id = measurements.user_id
            and tcm.member_user_id = auth.uid()
        )
      )
  )
  or exists (
    select 1 from secret_giver_requests sg
```

```
    where sg.target_id = measurements.user_id
      and sg.category = measurements.category
      and sg.status = 'approved'
      and sg.approved_until > now()
      and sg.requester_id = auth.uid()
  )
);
```

size_labels

sql
KopiujEdytuj

```
create policy labels_owner_rw on size_labels
for all using (user_id = auth.uid())
with check (user_id = auth.uid());

create policy labels_shared_r on size_labels
for select using (
  user_id = auth.uid()
  or exists (
    select 1 from share_rules sr
    where sr.owner_id = size_labels.user_id
      and sr.category = size_labels.category
      and sr.scope = 'read'
      and (sr.expires_at is null or sr.expires_at > now())
      and (
        sr.target_user_id = auth.uid()
        or exists (
          select 1 from trust_circle_members tcm
          join trust_circles tc on tc.id = tcm.circle_id
          where tc.owner_id = size_labels.user_id
            and tcm.member_user_id = auth.uid()
        )
      )
  )
  or exists (
    select 1 from secret_giver_requests sg
    where sg.target_id = size_labels.user_id
      and sg.category = size_labels.category
      and sg.status = 'approved'
      and sg.approved_until > now()
      and sg.requester_id = auth.uid()
  )
);
```

measurement_history – tylko właściciel

sql

KopiujEdytuj

```
create policy mhist_owner_r on measurement_history
for select using (
  exists (select 1 from measurements m
          where m.id = measurement_history.measurement_id
            and m.user_id = auth.uid())
);
```

A.4. Marki i mapy — publiczny odczyt (zalogowani)

sql

KopiujEdytuj

```
create policy brands_read on brands
for select using (true);

create policy bsm_read on brand_size_maps
for select using (true);

create policy beq_read on brand_equivalences
for select using (true);
```

A.5. Kręgi i reguły

sql

KopiujEdytuj

```
-- Krąg: właściciel czyta/edytuje swój krąg
create policy tc_owner_rw on trust_circles
for all using (owner_id = auth.uid())
with check (owner_id = auth.uid());

-- Członkowie: właściciel widzi, członek widzi siebie
create policy tcm_owner_r on trust_circle_members
for select using (
  exists (select 1 from trust_circles tc
          where tc.id = trust_circle_members.circle_id
            and tc.owner_id = auth.uid())
  or member_user_id = auth.uid()
);
create policy tcm_owner_w on trust_circle_members
for all using (
  exists (select 1 from trust_circles tc
          where tc.id = trust_circle_members.circle_id
            and tc.owner_id = auth.uid())
) with check (
  exists (select 1 from trust_circles tc
          where tc.id = trust_circle_members.circle_id
            and tc.owner_id = auth.uid())
```

```
);

-- Reguły udostępnień: właściciel zarządza; adresaci mogą je odczytać (dla UI)
create policy share_owner_rw on share_rules
for all using (owner_id = auth.uid())
with check (owner_id = auth.uid());

create policy share_target_r on share_rules
for select using (
  target_user_id = auth.uid()
  or exists (
    select 1 from trust_circle_members tcm
    join trust_circles tc on tc.id = tcm.circle_id
    where share_rules.circle_id = tc.id
      and tcm.member_user_id = auth.uid()
  )
);
```

A.6. Prezentowy link (tylko właściciel przez UI; publiczny dostęp przez Edge)

sql
KopiujEdytuj

```
-- Właściciel: odczyt/zarządzanie swoimi linkami
create policy sl_owner_rw on shared_links
for all using (owner_id = auth.uid())
with check (owner_id = auth.uid());

-- Brak publicznego selectu – render linku robi Edge Function z service role.
```

A.7. Secret Giver

sql
KopiujEdytuj

```
-- Requester i Target widzą swoje rekordy; edycja tylko przez Edge (statusy)
create policy sg_rt_r on secret_giver_requests
for select using (requester_id = auth.uid() or target_id = auth.uid());

-- Zmiany statusów i approved_until wykonujemy z Edge (service role) – brak polityk UPDATE dla klienta.
```

A.8. Pozostałe tabele (skróty)

sql
KopiujEdytuj

```
-- privacy_blocks, events, wishlists, wishlist_items
create policy pb_owner_rw on privacy_blocks
```

```
for all using (owner_id = auth.uid())
with check (owner_id = auth.uid());

create policy events_owner_rw on events
for all using (user_id = auth.uid())
with check (user_id = auth.uid());

create policy wl_owner_rw on wishlists
for all using (user_id = auth.uid())
with check (user_id = auth.uid());

create policy wli_owner_rw on wishlist_items
for all using (
  exists (select 1 from wishlists w where w.id = wishlist_items.wishlist_id and
w.user_id = auth.uid())
) with check (
  exists (select 1 from wishlists w where w.id = wishlist_items.wishlist_id and
w.user_id = auth.uid())
);

-- missions: global read
create policy missions_read on missions
for select using (true);

-- mission_progress: właściciel
create policy mprog_owner_rw on mission_progress
for all using (user_id = auth.uid())
with check (user_id = auth.uid());

-- notification_*: właściciel
create policy nr_owner_rw on notification_rules
for all using (user_id = auth.uid())
with check (user_id = auth.uid());

create policy nl_owner_r on notification_log
for select using (user_id = auth.uid());

-- subscriptions: właściciel (odczyt), zapisy tylko z webhooka (service role)
create policy subs_owner_r on subscriptions
for select using (user_id = auth.uid());
```

B) Edge Functions — API (Supabase Functions, TypeScript)

Wszystkie funkcje wykonują walidację planu, limitów i parametrów.
Wysyłka maili: Postmark/SendGrid. Push: FCM WebPush.
Brzegowe operacje (statusy SG, prezentowe linki public) robią się **tylko** w Edge z kluczem serwisowym.
Rate-limit: proste api_throttle (user_id, endpoint, window_start, counter) + check (np. max 30/min).

B.0. Wspólne helpery (skróty)

- getUser(ctx) – dekoduje JWT i pobiera auth.uid.
- assertPlanLimits(user_id, feature) – np. limity Kręgu, zakres Prezentowego linku.
- genToken(len=22) – losowy token (Base62).
- hashPassword(bcrypt/argon2) – dla linków Premium+.
- notify(user_id, {title, body, channel}) – adapter push/e-mail.

B.1. Prezentowy link

1) create_shared_link

POST /api/shared-links/create
Input: { scope, categories[], ttlDays?, maxViews?, oneTime?, password? }
Działanie:

- Sprawdza plan i limity (Free/Premium/Premium+).
- Ustala domyślny TTL wg planu (Free 7 / Premium 14 / Premium+ 30).
- Generuje token, opcjonalnie password_hash.
- Wstawia do shared_links i zwraca URL.

2) resolve_shared_link (publiczny)

GET /api/shared-links/{token}
Działanie:

- Ładuje rekord po token; sprawdza revoked_at, expires_at, max_views, one_time.
- Inkrementuje views_count (atomowo).
- Zwraca payload: wishlist_items + „bazowe rozmiary” (tylko zakres udostępniony).
- Jeżeli one_time=true i to pierwszy odczyt → od razu ustawia revoked_at=now().

3) revoke_shared_link

POST /api/shared-links/revoke
Input: { id } (właściciel) → revoked_at=now().

B.2. Secret Giver (token zawsze płatny)

1) sg_request_create_checkout

POST /api/sg/create-checkout
Input: { target_id, category, requested_hours }
Działanie:

- Tworzy **Stripe Checkout Session** (jednorazowa płatność 20 PLN).
- W metadata umieszcza requester_id, target_id, category, requested_hours.
- Zwraca checkout_url.

2) stripe_webhook (SG + subskrypcje + blokada)

POST /api/payments/stripe-webhook
Obsługa zdarzeń:

- checkout.session.completed (SG):
  - odczyt metadata ➜ tworzy secret_giver_requests z status='pending', wysyła powiadomienie do target_id.
- invoice.payment_succeeded / customer.subscription.updated:
  - aktualizuje subscriptions i pole plan w profiles.
  - Jeśli plan premium_plus stał się aktywny → w privacy_blocks wstawia/aktualizuje wpis feature='secret_giver', active_until = null (w cenie).
- (Opcjonalnie) payment_intent.succeeded dla blokady SG w Free/Premium (5 PLN/m) — jeżeli realizowane jako subskrypcja/produkt Stripe.

3) sg_request_approve_deny

POST /api/sg/resolve
Input: { request_id, action: 'approve'|'deny' }
Działanie (tylko target_id):

- approve → status='approved', approved_until = now() + requested_hours * interval '1 hour', notyfikacja do requester_id.

- deny → status='denied', notyfikacja do requester_id.

4) sg_access_guard

GET /api/sg/access?owner_id=&category=
Zwraca, czy **auth.uid()** ma aktywny dostęp SG do danej kategorii (pomocnicze dla UI).

Dostęp do danych właściwych (measurements/labels) i tak kontroluje RLS; guard służy tylko do wyświetlenia odpowiedniego stanu UI.

B.3. Krąg i reguły udostępniania

1) circles_invite

POST /api/circles/invite
Input: { circle_id, email }

- Sprawdza limit planu: Free=1, Premium=4, Premium+=∞.
- Wysyła mail z zaproszeniem / linkiem do rejestracji.
- Wstawia rekord do trust_circle_members po akceptacji (oddzielny endpoint accept_invite).

2) share_rules_set

POST /api/share-rules/set
Input: { target_user_id? | circle_id?, category, expires_at? }

- Tworzy/aktualizuje rekord w share_rules (właściciel).
- Waliduje, że tylko **per kategoria** i że target_user_id XOR circle_id.

B.4. Konwersje między markami

conversions_recommend

POST /api/conversions/recommend
Input: { brand_from?, label?, measurements?, brand_to?, category, target_gender }
Działanie:

- Tryb 1 (metka): brand_from + label → odczyt zakresów z brand_size_maps → projekcja na brand_to (lub wszystkie).
- Tryb 2 (pomiary): dopasowanie zakresów dla każdej marki.
- Zwraca listę { brand_id, label_value, confidence }.

Endpoint czyta tylko tabele marek/map (publiczne w RLS).

B.5. Powiadomienia i digest

1) compute_notifications_daily

CRON (07:30 lokalnie):

- Reguły: „okazje” (21/7/3/1), „stare wymiary” (12/24 mies.), dzieci (co 3 mies.), wygasanie linków (72/48/24 h), misje prawie ukończone.
- Tworzy wpisy w notification_log i wywołuje notify() zgodnie z opt-in i godzinami ciszy.

2) compute_digest_weekly

CRON (pon 08:00):

- Agreguje tygodniową aktywność (liczniki) i wysyła krótki mail/push.

B.6. Misje

1) missions_progress_event

POST /api/missions/progress
Input: { code, delta? } – idempotentny zapis postępu (np. event „konwersja wykonana”).

- Aktualizuje mission_progress.
- Jeśli warunek spełniony → ustawia completed_at i wysyła notyfikację „Odbierz nagrodę”.

2) missions_claim_reward

POST /api/missions/claim
Input: { mission_id }

- Generuje kupon Stripe / dodaje privacy_blocks (+X dni blokady SG) lub zapisuje „badge”.
- reward_claimed_at = now().

C) Dodatkowe mechanizmy bezpieczeństwa

1. **Rate limiting** (tabela api_throttle lub Supabase Rate Limits na funkcjach):
    - shared-links/create: max 10/h; resolve: max 60/min z IP.
    - sg/create-checkout: max 3/h per user.
    - sg/resolve: max 20/h per user.
2. **Audit**: logujemy w notification_log / dodatkowo audit_log (opcjonalna tabela: user_id, action, object, meta, at).
3. **Revocation everywhere**: linki i dostępy SG można unieważnić natychmiast; UI musi mieć wyraźny przycisk **REVOKE**.
4. **Domyślna prywatność**: wszystko prywatne, udostępnienia jawne, granularne, z datą wygaśnięcia (domyślnie **brak** wygaśnięcia, ale rekomenduję 12 mies. dla share_rules jako „miękki bezpiecznik").
5. **Hasła do linków (Premium+)**: przechowujemy **hash** (Argon2), porównanie po stronie Edge, nigdy czysty tekst.
6. **Idempotencja webhooków Stripe**: używamy idempotency_key i tabeli stripe_events (opcjonalnie) by nie przetwarzać zdarzeń dwa razy.

D) Minimalne kontrakty (TypeScript szkice)

Supabase Edge Function (deno/node), tu pseudo-TS dla czytelności.

ts
KopiujEdytuj
```
// shared-links/create
export default async function handler(req) {
  const { userId } = authFromHeader(req);           // wymuszamy zalogowanie
  const body = await req.json();
  const plan = await db.selectOne('profiles', { user_id: userId }, ['plan']);

  const { scope, categories, ttlDays, maxViews, oneTime, password } =
validateCreateLink(body, plan);
  const token = genToken(22);
  const expires_at = calcExpiry(ttlDays ?? defaultTTL(plan)); // Free=7,
Premium=14, Premium+=30
  const password_hash = plan === 'premium_plus' && password ? await
hashPassword(password) : null;

  await db.insert('shared_links', {
```

```
    owner_id: userId, scope, categories, token, password_hash,
    one_time: !!oneTime,
    max_views: capMaxViews(maxViews, plan),
    expires_at
  });

  return json({ url: `https://sizesync.link/${token}`, expires_at });
}
ts
KopiujEdytuj
// sg/create-checkout
export default async function handler(req) {
  const { userId } = authFromHeader(req);
  const { target_id, category, requested_hours } = await req.json();
  assertNotBlockedSG(target_id); // sprawdź privacy_blocks targeta (blokada SG)
  const session = await stripe.checkout.sessions.create({
    mode: 'payment',
    line_items: [{ price: PRICE_SG_TOKEN_PLN_20, quantity: 1 }],
    success_url: APP_URL + '/sg/success',
    cancel_url: APP_URL + '/sg/cancel',
    metadata: { requester_id: userId, target_id, category, requested_hours }
  });
  return json({ url: session.url });
}
ts
KopiujEdytuj
// payments/stripe-webhook (fragment SG)
if (event.type === 'checkout.session.completed') {
  const s = event.data.object as Stripe.Checkout.Session;
  if (s.metadata?.target_id && s.metadata?.requester_id) {
    await db.insert('secret_giver_requests', {
      requester_id: s.metadata.requester_id,
      target_id: s.metadata.target_id,
      category: s.metadata.category,
      status: 'pending',
      requested_hours: Number(s.metadata.requested_hours) || 48
    });
    await notify(s.metadata.target_id, {
      title: 'Prośba o dostęp do rozmiarów',
      body: 'Masz nową prośbę Secret Giver — zaakceptuj lub odrzuć.'
    });
  }
}
```

Dodatkowa tabela: stripe_events (idempotencja webhooków)

sql
KopiujEdytuj

```
create table stripe_events (
  id uuid primary key default gen_random_uuid(),
  event_id text not null unique,          -- np. 'evt_1Q...'
  event_type text not null,               -- np. 'checkout.session.completed'
  payload jsonb not null,                 -- cały event (zredagowany z PII, jeśli potrzebne)
  received_at timestamptz not null default now(),
  processed_at timestamptz,               -- null = jeszcze nie przetworzony
  processing_error text                   -- ostatni błąd, jeśli był
);

create index idx_stripe_events_type on stripe_events(event_type);
create index idx_stripe_events_processed on stripe_events(processed_at);
```

Wzorzec użycia w webhooku:

1. Upsert po event_id.
2. Jeśli processed_at ≠ null → **kończ** (idempotencja).
3. Przetwórz zdarzenie → ustaw processed_at=now() (lub processing_error).

Moduł 4 — Komponenty UI (MVP PWA)

Poniżej lista ekranów, kluczowych widoków, stanów (loading/empty/error/offline), przepływów oraz zestaw komponentów z krótkimi wskazówkami implementacyjnymi (React + Next.js + Tailwind + Headless UI). Dostosowane do light/dark i PL/EN.

4.1. Informacja architektoniczna (frontend)

- **Routing**: Next.js App Router (/app) + segmenty chronione ((auth), (protected)).
- **Stan**: React Query (server state), Zustand (UI/local), i18next (PL/EN).
- **Offline**: Service Worker + Dexie/IndexedDB (cache modelu: profil, ZPR, marki, linki, misje).
- **A11y**: Headless UI (Dialog, Listbox, Menu), aria-labels + focus ring.

4.2. Mapowanie ekranów

A. Autoryzacja i onboarding

- **/auth/login**: logowanie (e-mail, Google/Apple).
    - Stany: loading, błąd, „zapamiętaj mnie”.
- **/onboarding**: samouczek (slajdy) + szybka konfiguracja: język, jednostki, powiadomienia, motyw (domyślnie „system”).
    - **CTA** do: „Dodaj wymiary” lub „Dodaj metkę”.

B. Home (hub)

- **/home**: karty skrótów: „Dodaj wymiary”, „Dodaj metkę”, „Konwersje”, „Utwórz Prezentowy link”.
    - Sekcje: „Nadchodzące okazje”, „Misje tygodnia”, „Ostatnie aktywności Kręgu”.
    - Offline badge, jeśli brak sieci (ikonka + podpowiedź).

C. ZPR (profil rozmiarów)

- **/profile**: zakładki/karty kategorii: Odzież (tops/bottoms/outerwear/underwear), Buty, Akcesoria, Biżuteria.
    - Widok kategorii: lista wymiarów (z jednostkami), przycisk „Edytuj”, „Historia zmian”, „Dodaj metkę”.
    - **Empty state**: ilustracja + „Zacznij od pomiarów / metki”.
- **/profile/measure**: formularz z instrukcjami (mini-rysunki, tooltipy).
- **/profile/label**: kreator „metki”: marka → kategoria → etykieta → wynik **Auto-konwersje** (lista marek + confidence).

D. Marki i konwersje

- **/brands**: lista marek (wyszukiwarka, filtry: kategoria/target/region).
- **/conversions**: ekran narzędziowy — wejście: metka **albo** wymiary → wyjście: tabela konwersji (z badge „official/derived”).
    - **Offline**: działa na cache, ostrzeżenie, jeśli brakuje danych dla danej marki.

E. Zaufany Krąg i udostępnienia

- **/circle**: lista członków (limit wg planu), zaproszenia, usuwanie.
- **/sharing**: reguły per kategoria (toggle „czytaj”), data wygaśnięcia (opcjonalnie), podgląd „co widzi adresat”.

F. Secret Giver

- **/sg/request**: formularz (adresat, kategoria, czas, podgląd prośby) → **Checkout Stripe** (token płatny zawsze).
- **/sg/inbox**: skrzynka odbiorcy (lista próśb) z akcjami **Akceptuj/Odrzuć**; banner o blokadzie SG (wg planu).
    - Stany: pending, approved (z zegarem do wygaśnięcia), denied/expired.
    - CTA po akceptacji: „Przejdź do profilu rozmiarów (kategoria)".

G. Prezentowy link

- **/links**: lista linków (status: aktywny/wygasły/jednorazowy), licznik odsłon, przyciski **Kopiuj**/**Revoke**/**Przedłuż**.
- **/links/create**: kreator — scope, kategorie, TTL (wg planu), max_views (wg planu), one_time, (Premium+: hasło).
- **/s/{token}** (public): podgląd gościa — wishlist + bazowe rozmiary, licznik; obsługa hasła (Premium+).

H. Okazje, wishlista, misje

- **/events**: kalendarz (miesiąc/lista), dodawanie okazji (urodziny, rocznice).
- **/wishlist**: lista życzeń (kafle: tytuł, link, kategoria, notatki).
- **/missions**: misje (once/weekly), pasek postępu, **Odbierz nagrodę** (kupon/bonus blokady SG).

I. Ustawienia

- **/settings**: konto, język, jednostki, **motyw (system/light/dark)**, powiadomienia (granularnie), plan/płatności, prywatność (SG blokada, linki aktywne), eksport/konto.

4.3. Inwentarz komponentów (Tailwind/HeadlessUI)

Nawigacja

- AppShell: nagłówek, bottom-nav (mobile) / side-nav (desktop).
- TabBar (HeadlessUI Tab.Group) – sekcje ZPR.
- Breadcrumbs dla ścieżek narzędziowych (konwersje, linki).

Formularze i wejścia

- Field (label + helper + error), NumberField z jednostkami (cm/in).
- Select (HeadlessUI Listbox) – marka, kategoria, rozmiar.

- RangeBadge – prezentacja zakresów z map marek.
- Toggle (HeadlessUI Switch) – powiadomienia, dark mode, one_time.

Karty i listy

- Card (rounded-2xl, shadow-sm), wariant ClickableCard.
- ListItem z ikoną/awatarem, akcje w Menu (Headless Menu).

Dialogi i toasty

- Modal (Headless Dialog) – zgody SG, potwierdzenia Revoke.
- Toast (portale) – sukces/błąd/ostrzeżenie (timeout + aria-live).

Stany

- EmptyState (ilustracja + CTA), ErrorState (z retry), OfflineBadge (PWA).
- LoadingSpinner (ARIA).

Komponenty specjalne

- ConversionTable – tabela konwersji z kolumnami: Marka / Region / Label / Confidence.
- LinkCard – status prezentowego linku (TTL badge, max_views progress).
- MissionTile – kafelek misji z paskiem postępu i przyciskiem „Odbierz".
- CircleMember + ShareRuleRow.
- SGRequestItem (status, czas do wygaśnięcia, CTA).

4.4. Przepływy kluczowe (UX bez tarcia)

1) Dodaj metkę → Auto-konwersje

- Szybki kreator (3 kroki, max 2 kliknięcia na krok).
- Po wyborze etykiety: preload panelu wyników; pokaż top 5 marek + „Zobacz wszystkie".
- Badge „Źródło: oficjalne / pochodne", mini-tooltip „co to znaczy".

2) Tworzenie Prezentowego linku

- Ekran z presetami wg planu (Free/Premium/Premium+).
- Live-preview: na kartce obok widać, co gość zobaczy (wishlist + rozmiary).

- Po stworzeniu: toast „Skopiowano link”, przycisk „Udostępnij…”.

3) Prośba SG (checkout)

- Formularz minimalny: użytkownik docelowy, kategoria, czas (select).
- Po „Kontynuuj” → Stripe Checkout (nowe okno/zakładka).
- Po sukcesie: redirect do /sg/inbox z banerem „Czekaj na zgodę”.

4) Zgoda SG (odbiorca)

- Powiadomienie → /sg/inbox → szczegóły prośby (kto/co/jak długo).
- „Akceptuj” → snackbar „Dostęp do [kategoria] do hh:mm, możesz cofnąć w każdej chwili”.
- Link „Zobacz co widzi darczyńca” (podgląd read-only).

5) Misje

- W misjach weekly pokaż „ile brakuje” (np. 1 akcja).
- Po zaliczeniu – confetti micro (preferowane dyskretne) + modal „Odbierz nagrodę”.
- Kupon Premium: kod i przycisk „Zastosuj teraz” (przenosi do płatności/Stripe Portal).

4.5. Stany, błędy, offline

- **Offline**: badge w headerze, disable akcji sieciowych, kolejka mutacji (ikona „czeka na synchronizację”).
- **Błędy**: kody przyjazne (np. LINK_EXPIRED, SG_BLOCKED) + CTA.
- **Empty states** dopracowane:
  - ZPR pusty → „Zacznij od pomiarów/metki” (2 przyciski).
  - Brak członków Kręgu → CTA „Zaproś pierwszą osobę” (+ korzyści).
  - Brak misji weekly → info kiedy reset.

4.6. Design tokens (Tailwind)

```
js
KopiujEdytuj
// tailwind.config.js – skrót idei
theme: {
  extend: {
```

```
    colors: {
      bg: { light: '#F7F7F7', dark: '#111315' },
      text: { light: '#1F2937', dark: '#E7E7E7' },
      primary: { DEFAULT: '#4C5B5C' },
      accent: { DEFAULT: '#E07A5F' },
      success: { DEFAULT: '#81B29A' },
      border: { light: '#E5E7EB', dark: '#2A2F33' },
    },
    borderRadius: { 'xl2': '1.25rem' },
    boxShadow: { soft: '0 8px 24px rgba(0,0,0,.06)' },
  }
}
```

Aplikowanie motywu: class="dark" na <html> (auto z preferencją systemową + przełącznik w Ustawieniach). Kontrasty sprawdzamy axe.

4.7. i18n (PL/EN)

- Namespace'y: auth, onboarding, profile, labels, conversions, circle, sharing, sg, links, events, wishlist, missions, settings, common.
- Klucze krótkie, bez interpolacji HTML (używaj <Trans> w React tylko gdzie potrzebne).

4.8. Mikro-kopie (UX copy — przykłady)

- **SG prośba (powiadomienie):** „{name} prosi o dostęp do Twoich rozmiarów w kategorii **{category}** na **{hours} h**."
- **Link wygasa:** „Twój Prezentowy link wygaśnie za **{hours} h**. Przedłużyć?"
- **Konwersje:** „Na podstawie tabel marki **{brand}** i Twoich wymiarów."

4.9. Checklist UI (skrót pod QA)

- Focus trap w modalach, ESC zamyka, aria-* kompletne.
- Każda akcja ma stan loading (disabled + spinner).
- Długie listy: wirtualizacja (react-virtualized) dla marek/map.
- Responsywność: mobile first, ≥320px; desktop używa side-nav.
- Ikony: lucide-react (lekki, spójny zestaw).

5.1 Zasady ogólne QA

- **Środowiska:** dev → staging (kopie konfiguracji prod, fake płatności) → prod.
- **Kontrolki wejścia:** feature flags (misje, digest, SG), włączone logi Edge.
- **Definicje powagi błędu:**
  P0 – krytyczny (dane/bezpieczeństwo/blokada płatności) → blokuje release.
  P1 – wysoka (główne flow nie działa lub psuje dane) → blokuje release.
  P2 – średnia (obejście możliwe) → naprawa w najbliższym patchu.
  P3 – niska (kosmetyka) → może iść na backlog.

5.2 Zestaw kont testowych i dane startowe

- Konta: free.user@…, premium.user@…, plus.user@…, sg.requester@…, sg.target@….
- Profile: PL i EN; metryczne i imperialne; 1 profil „dziecko”.
- Seed marek: min. 6 marek (EU/US/UK), po 2–3 kategorie.
- Events: urodziny za 21/7/3/1 dni; 2 rocznice.
- Wishlista: po 5 pozycji na konto.
- Misje: ONBOARD_1/2, WEEKLY_ZPR/CONV/LINK aktywne.

5.3 Testy funkcjonalne (by feature)

A) Autoryzacja i profil

- Rejestracja e-mail/hasło + OAuth (Google/Apple) – poprawna nawigacja do onboarding.
- Zmiana języka/jednostek/motywu → trwałe po reload.
- Plan (Free/Premium/Plus) widoczny w UI; zmiana planu odświeża limity.

B) ZPR (pomiary + metka)

- Dodanie/edycja pomiarów (cm/in) → poprawna konwersja, zapis w DB.
- Historia tworzy się przy każdej zmianie; przywrócenie poprzedniej wersji.
- Dodanie „metki” (brand/kategoria/etykieta) → natychmiastowa **auto-konwersja** (z confidence).

- Brak metadanych marki → komunikat i ścieżka obejścia (dodanie ręczne innej marki).

C) Konwersje między markami

- Tryb METKA: A/label → sugerowane B/C/D; zmiana regionu (EU/US/UK) zmienia tabelę.
- Tryb POMIARY: dopasowanie do marek; brak dopasowania → informacja + najbliższe zakresy.
- Offline cache: działa dla wcześniej pobranych marek; jasny komunikat gdy brak danych.

D) Zaufany Krąg i udostępnienia

- Free: limit 1 osoba; Premium: 4; Plus: bez limitu.
- Ustawienie **per kategoria** (read) – członek widzi tylko to, co udostępniono.
- Wygaśnięcie reguły → automatyczna utrata dostępu (czas testowy skrócony).

E) Prezentowy link

- Kreator: TTL i limity zgodne z planem; **one_time** i (Plus) **hasło**.
- Widok publiczny /s/{token}: działa bez logowania; one_time wygasza po 1. otwarciu.
- max_views i expires_at – egzekwowane, licznik rośnie; **REVOKE** działa natychmiast.
- Log dostępu (anonimowy) – licznik i timestampy (zgodnie z planem).

F) Secret Giver (token płatny)

- sg.requester tworzy Checkout (20 PLN) → po sukcesie powstaje **pending** request.
- sg.target dostaje powiadomienie, może **approve/deny**; po akceptacji clock do approved_until.
- Odczyt danych przez sg.requester tylko w wybranej kategorii i tylko do wygaśnięcia.
- **Blokada SG:** Free/Premium – zakup blokady działa; Plus – blokada w cenie.

G) Okazje, wishlista, misje

- Dodanie/edycja/usuwanie eventów; widok listy/miesiąca.

- Wishlista – dodaj link, brak miniatury ≠ błąd; sortowanie i filtrowanie.
- Misje: widok postępu, zdarzenia zaliczają progres (np. konwersja=+1), **Odbierz nagrodę** działa (kupon, dni blokady).

H) Powiadomienia

- Konfigurowalne per kategoria; **godziny ciszy** działają.
- Scenariusze: okazje (21/7/3/1), stare wymiary (12/24 mies.), dzieci (co 3 mies.), link expiry (72/48/24h), SG (natychmiast/24h), misje (max 1/d).
- Tygodniowy digest w pon. 08:00 – poprawne treści i liczby.

5.4 Testy bezpieczeństwa i RLS

RLS (dostęp do danych)

- **Właściciel**: pełny odczyt/edycja swoich measurements, size_labels, history.
- **Członek Kręgu**: widzi tylko kategorie z **share_rules**; brak „przecieku" do innych kategorii.
- **SG requester**: dostęp tylko **approved** + aktywne, tylko kategoria; po wygaśnięciu 403.
- **Prezentowy link**: brak SELECT po stronie klienta; tylko Edge może odczytać payload.
- Próba „enumeracji" cudzych ID → 403 (w każdej tabeli prywatnej).

Edge/Payments

- Webhook Stripe idempotentny (tabela stripe_events); drugi raz event → ignorowany.
- Zmiana planu (Stripe/RevenueCat) aktualizuje profiles.plan i blokadę SG w Plus.
- SG blokada nie wyłącza istniejących, **zatwierdzonych** dostępów (dotyczy tylko nowych) – komunikat w UI.

Rate limiting

- shared-links/create > 10/h → błąd 429 i jasny komunikat.
- sg/create-checkout > 3/h → 429.
- Public /s/{token} > 60/min/IP → 429 (ochrona przed scrapem).

5.5 Testy offline i synchronizacja

- Brak sieci:
    - Edycja ZPR zapisuje w kolejce; po „powrocie” synchronizuje (konflikt → *last-write-wins* + banner „nadpisano starsze dane”).
    - Konwersje działają na cache; dla brakujących marek — komunikat i przycisk „Pobierz przy połączeniu”.
    - Operacje sieciowe (SG, linki) są wyłączone z wyraźnym stanem disabled + tooltip.

5.6 Testy wydajności i UX

- **TTI PWA** < 2,5 s na 3G Fast (Lighthouse).
- **Bundle** < ~250–300 kB JS na start (kod split/SSR).
- Listy marek/map: wirtualizacja zachowuje płynność przy 5–10k wierszy.
- Scroll i interakcje 60 fps na mid-range telefonie.

5.7 Testy dostępności (WCAG 2.1 AA)

- Kontrast tekstu/CTA, focus-visible na wszystkich interaktywnych.
- Pełna nawigacja klawiaturą (modale, menu, listy).
- Etykiety aria-* dla ikon, liczników (np. odliczanie SG).
- Czytniki ekranu: tytuły stron, role dialogów, opisy przycisków.
- Animacje zredukowane przy prefers-reduced-motion.

5.8 Testy i18n / l10n

- Przełącznik PL/EN natychmiast zmienia treści; brak twardo zakodowanych stringów.
- Jednostki: cm/in niezależnie od języka; formaty dat (PL: DD.MM, EN: MMM D).
- Teksty powiadomień i maili w obu językach (placeholders działają).

5.9 Testy prywatności i RODO

- Eksport konta (JSON) + „Usuń konto” → pełna anonimizacja/soft-delete.
- Shared Links: domyślnie krótki TTL, możliwość **REVOKE** w każdej chwili.
- Notification log i access logi trzymane zgodnie z retencją (90 dni, potem agregat).
- DPIA checklist przechodzimy bez zastrzeżeń (minimalizacja danych, celowość, opt-in).

5.10 Testy zgodności przeglądarek/ urządzeń

- Chrome, Safari, Firefox (ostatnie 2 wersje). iOS Safari, Android Chrome.
- PWA install: iOS/Android/desktop — splash, ikonka, offline.
- Dark/Light mode: automatyka + przełącznik ręczny (bez „mrugnięć”).

5.11 E2E scenariusze (Playwright – propozycje)

1. **Onboarding → ZPR → Konwersje**
   Rejestracja → wprowadź 3 kategorie → dodaj metkę → zobacz propozycje marek.
2. **Free → Prezentowy link**
   Stwórz link (TTL 7, max_views 5, one_time off) → otwórz publicznie 5× → 6. raz = blocked → REVOKE.
3. **Krąg i udostępnienie**
   Free: dodaj 1 członka → udostępnij „Buty” → zaloguj jako członek → widzisz tylko „Buty”.
4. **SG – płatny token**
   Requester: Checkout 20 PLN → target: approve 48 h → requester widzi tylko „Biżuteria” → po 48 h 403.
5. **Misje i nagroda**
   Ukończ ONBOARD_1 → odbierz „7 dni blokady SG” → wpis w privacy_blocks.
6. **Powiadomienia**
   Zasymuluj CRON: event za 3 dni → push + e-mail; link expiry w 24 h → push z przyciskiem „Przedłuż”.

5.12 Kryteria wydania (Go/No-Go)

- Brak **P0/P1** w staging.
- RLS testy przechodzą (checklista A/B/C/D z pkt 5.4).
- Lighthouse: Performance ≥ 80, Accessibility ≥ 90.
- E2E komplet przechodzi (6 scenariuszy).
- Webhook Stripe – 100% idempotencji w stripe_events.
- Powiadomienia wysyłają się zgodnie z opt-in i godzinami ciszy.

5.13 Operacja po wydaniu (post-release)

- Monitor: błędy Edge, 4xx/5xx, Web Vitals, wskaźniki retencji.
- Hotfix SLA: P0 ≤ 24h, P1 ≤ 72h.
- Pętle feedbacku: formularz w aplikacji + triage co tydzień.

1) Struktura repo (propozycja)

bash
KopiujEdytuj

```
/apps/web          # Next.js PWA
/apps/edge         # Supabase Edge Functions (TypeScript)
/packages/db       # SQL DDL + seedy + RLS
/packages/tests    # Playwright + fixtures
```

2) Pliki konfiguracyjne

2.1 .env.local (web)

bash
KopiujEdytuj

```
NEXT_PUBLIC_SUPABASE_URL=https://<YOUR-PROJECT>.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=<anon-key>
NEXT_PUBLIC_APP_URL=http://localhost:3000
NEXT_PUBLIC_FCM_VAPID_PUBLIC_KEY=<web-push-key>

STRIPE_SECRET_KEY=sk_test_xxx
STRIPE_PRICE_SG_TOKEN=price_20PLN_xxx
```

STRIPE_WEBHOOK_SECRET=whsec_xxx

POSTMARK_TOKEN=xxx # albo SENDGRID_API_KEY=xxx

2.2 .env (edge)

bash
KopiujEdytuj
SUPABASE_URL=https://<YOUR-PROJECT>.supabase.co
SUPABASE_SERVICE_ROLE_KEY=<service-role-key>
APP_URL=http://localhost:3000
STRIPE_SECRET_KEY=sk_test_xxx
STRIPE_PRICE_SG_TOKEN=price_20PLN_xxx
STRIPE_WEBHOOK_SECRET=whsec_xxx
POSTMARK_TOKEN=xxx

3) Seedy SQL (skrót — wklej kolejno w Supabase SQL Editor)

3.1 Seed marek i map (przykładowe)

sql
KopiujEdytuj
```
insert into brands (id,name,country,website) values
  (gen_random_uuid(),'Acme Apparel','PL','https://acme.pl'),
  (gen_random_uuid(),'North River','US','https://nriver.com'),
  (gen_random_uuid(),'Urban Line','UK','https://urbanline.uk');

-- buty EU (wartości przykładowe, demonstracyjne)
with b as (
  select id from brands where name='Acme Apparel' limit 1
)
insert into brand_size_maps
(brand_id,category,target,region,label_value,min_foot_len_cm,max_foot_len_cm,so
urce,version)
select id,'shoes','unisex','EU','42',26.0,26.6,'official',1 from b
union all
select id,'shoes','unisex','EU','43',26.7,27.3,'official',1 from b;

-- t-shirty (klatka/biust)
with b as (select id from brands where name='North River')
insert into brand_size_maps
(brand_id,category,target,region,label_value,min_chest_cm,max_chest_cm,source,
version)
select id,'tops','men','US','M',96,101,'official',1 from b
```

union all
select id,'tops','men','US','L',102,106,'official',1 from b;

3.2 Konta testowe (profile)

Użytkowników tworzysz w Supabase Auth (Email OTP lub „Invite user”). Po utworzeniu — dodać rekordy profilu:

sql
KopiujEdytuj

```
-- załóż, że znasz user_id z auth.users
insert into profiles (user_id,display_name,locale,plan,unit_preference)
values
  ('<free_user_id>','Free User','pl-PL','free','metric'),
  ('<premium_user_id>','Premium User','pl-PL','premium','metric'),
  ('<plus_user_id>','Plus User','en-US','premium_plus','metric'),
  ('<sg_requester_id>','SG Requester','pl-PL','free','metric'),
  ('<sg_target_id>','SG Target','pl-PL','free','metric');
```

3.3 ZPR i przykładowe metki

sql
KopiujEdytuj

```
-- ZPR Premium User
insert into measurements
(user_id,category,chest_cm,waist_cm,hip_cm,foot_len_cm)
values
  ('<premium_user_id>','tops',98,84,100,null),
  ('<premium_user_id>','shoes',null,null,null,26.4);

-- metka (buty EU 42) dla Premium User
insert into size_labels
(user_id,brand_name,category,label_value,region,confidence)
values ('<premium_user_id>','Acme Apparel','shoes','42','EU',95);
```

3.4 Okazje + wishlista

sql
KopiujEdytuj

```
insert into events (user_id,title,event_date,repeat_yearly) values
('<premium_user_id>','Urodziny Ani', current_date + interval '21 days', true),
('<premium_user_id>','Rocznica', current_date + interval '7 days', true);

insert into wishlists (id,user_id,name) values
(gen_random_uuid(),'<premium_user_id>','Lista życzeń');
insert into wishlist_items (wishlist_id,title,url,category)
select id,'Bransoletka srebrna','https://shop.example/item/1','jewelry' from wishlists
where user_id='<premium_user_id>' limit 1;
```

3.5 Misje startowe

sql
KopiujEdytuj
```
insert into missions (code,title,description,type,reward,reward_payload)
values
('ONBOARD_1','Start','Uzupełnij ≥3 kategorie w
ZPR','once','sg_block_days','{"days":7}'),
('WEEKLY_LINK','Link tygodnia','Utwórz Prezentowy
link','weekly','sg_block_days','{"days":7}');
```

4) Mock Stripe i webhook

4.1 Uruchom webhook lokalnie

bash
KopiujEdytuj
```
stripe listen --forward-to localhost:54321/functions/v1/payments/stripe-webhook
```

4.2 Symulacja zakupu tokenu SG (20 PLN)

bash
KopiujEdytuj
```
stripe trigger checkout.session.completed \
  --add charge:amount=2000 \
  --add checkout.session:mode=payment \
  --add checkout.session:metadata[requester_id]=<sg_requester_id> \
  --add checkout.session:metadata[target_id]=<sg_target_id> \
  --add checkout.session:metadata[category]=jewelry \
  --add checkout.session:metadata[requested_hours]=48
```

Sprawdź, czy powstał rekord w secret_giver_requests (pending) i czy sg_target dostał powiadomienie.

5) Symulacje CRON (Edge Functions)

5.1 Odpal dzienne notyfikacje

bash
KopiujEdytuj
```
curl -X POST "http://localhost:54321/functions/v1/compute-notifications-daily" \
 -H "Authorization: Bearer $SUPABASE_SERVICE_ROLE_KEY"
```

5.2 Odpal tygodniowy digest

bash
KopiujEdytuj

```
curl -X POST "http://localhost:54321/functions/v1/compute-digest-weekly" \
 -H "Authorization: Bearer $SUPABASE_SERVICE_ROLE_KEY"
```

6) Playwright — scenariusze E2E (skróty)

6.1 Konfiguracja

/packages/tests/playwright.config.ts

ts
KopiujEdytuj
```
import { defineConfig } from '@playwright/test';
export default defineConfig({
  use: {
    baseURL: process.env.BASE_URL || 'http://localhost:3000',
    locale: 'pl-PL',
    storageState: 'storage/free.json'  // zapisane sesje dla kont testowych
  },
  timeout: 60_000
});
```

6.2 Helper: logowanie szybką ścieżką (jeśli bez SSO)

ts
KopiujEdytuj
```
export async function loginAs(page, email: string, password: string) {
  await page.goto('/auth/login');
  await page.getByLabel('Email').fill(email);
  await page.getByLabel('Hasło').fill(password);
  await page.getByRole('button', { name: 'Zaloguj' }).click();
  await page.waitForURL('**/home');
}
```

6.3 Test: Free → Prezentowy link (TTL/limit/revoke)

ts
KopiujEdytuj
```
import { test, expect } from '@playwright/test';
test('Free: prezentowy link TTL/limit/revoke', async ({ page }) => {
  await loginAs(page, 'free.user@example.com', 'Passw0rd!');
  await page.goto('/links/create');
  await page.getByRole('button', { name: 'Utwórz' }).click();
  const url = await page.getByTestId('created-link-url').innerText();

  // 5 wejść OK, 6-te blokada
  for (let i=0; i<5; i++) {
    const p = await page.context().newPage();
```

```
    await p.goto(url);
    await p.close();
  }
  const p6 = await page.context().newPage();
  await p6.goto(url);
  await expect(p6.getByText(/Link wygasłlLimit odsłon/)).toBeVisible();
  await p6.close();

  // Revoke
  await page.goto('/links');
  await page.getByRole('button', { name: /RevokelUnieważnij/ }).first().click();
  await expect(page.getByText(/Unieważniono/)).toBeVisible();
});
```

6.4 Test: SG — płatny token → approve → wygaśnięcie

ts
KopiujEdytuj

```
test('SG: płatny token -> approve -> expiry', async ({ page, context }) => {
  // Requester tworzy i "opłaca" (tu zakładamy, że webhook już utworzył pending)
  await loginAs(page, 'sg.requester@example.com', 'Passw0rd!');
  await page.goto('/sg/request');
  // ... klik do checkoutu pomijamy w E2E; webhook seeduje pending ...

  // Target akceptuje
  const t = await context.newPage();
  await loginAs(t, 'sg.target@example.com', 'Passw0rd!');
  await t.goto('/sg/inbox');
  await t.getByRole('button', { name: /Akceptuj/ }).first().click();
  await expect(t.getByText(/Dostęp do/)).toBeVisible();

  // Requester widzi tylko daną kategorię
  await page.goto('/profile');
  await expect(page.getByText(/Biżuteria/)).toBeVisible();

  // Szybkie "wygaśnięcie": backend w trybie test skraca approved_until (flag)
  // Po wygaśnięciu:
  await page.reload();
  await expect(page.getByText(/Dostęp wygasłlBrak uprawnień/)).toBeVisible();
});
```

6.5 Test: Krąg i udostępnienia per kategoria

ts
KopiujEdytuj

```
test('Krąg: limity i share per kategoria', async ({ page, context }) => {
  await loginAs(page, 'premium.user@example.com', 'Passw0rd!');
```

```
  await page.goto('/circle');
  // Dodaj 4 osoby -> 5-ta powinna się nie udać
  for (let i=0; i<4; i++) { /* dodaj zaproszenie */ }
  // share per kategoria
  await page.goto('/sharing');
  // ustaw "shoes" dla jednego członka
  // Zaloguj się jako członek i sprawdź widoczność tylko "shoes"
});
```

7) NPM skrypty (dev quality-of-life)

Web (Next.js)

json
KopiujEdytuj

```
{
  "scripts": {
    "dev": "next dev",
    "build": "next build && next export",
    "lint": "eslint .",
    "typecheck": "tsc --noEmit",
    "test:e2e": "playwright test",
    "test:e2e:ui": "playwright test --ui"
  }
}
```

Edge

json
KopiujEdytuj

```
{
  "scripts": {
    "dev": "supabase functions serve",
    "deploy": "supabase functions deploy --project-ref <ref> --no-verify-jwt payments/
stripe-webhook compute-notifications-daily compute-digest-weekly",
    "lint": "eslint ."
  }
}
```

8) Szybkie checklisty „ręczne”

- **Link publiczny**: stwórz (Free), otwórz 5×, 6× blokada, REVOKE, potem 404/ expired.
- **Konwersje**: dodaj metkę „Acme Apparel EU 42” → zobacz propozycje innych marek.
- **Offline**: wyłącz sieć → edytuj ZPR → włącz sieć → sprawdź synchronizację.

- **Powiadomienia**: odpal CRON, zobacz push/mail PL i EN.
- **SG blokada**: w Free brak blokady → wykup 5 PLN/m → nowa prośba SG ma status „zablokowano”.