Målvaktsstatistik – README

En liten, fristående webapp för att registrera och analysera målvaktsstatistik med Supabase som backend. Appen består av tre sidor:

- `index.html` – dagligt arbete: lägg till match, klicka in skott, se listor/grafer, export/import.
- `compare.html` – välj 2–4 målvakter i samma team och jämför över tid.
- `goalie.html` – detaljer för en enskild målvakt (lista + grafer + export).

Arkitekturen är helt statisk (HTML/JS/CSS) + Supabase (Postgres, RLS, Auth). Ingen egen server krävs.

Quick start

1) **Skapa ett Supabase-projekt** och aktivera **Email/Password** under *Auth* → *Providers*. 2) **Lägg in databas-schemat** (tabeller, funktion, policies) i SQL Editor (se avsnitt *Databas & RLS* nedan). 3) I projektet, öppna `index.html`, `compare.html`, `goalie.html` och sätt: - `SUPABASE_URL` = ditt projekt-URL - `SUPABASE_ANON` = ditt anon key 4) Lägg till din lokala origo-domän i *Auth* → *URL Configuration* → *Redirect URLs* (t.ex. `http://localhost:5500` eller din deploy-domän). 5) Öppna `index.html` i webbläsaren. **Skapa konto** → **Logga in** → appen seed:ar målvakter för ditt team.

Projektstruktur

```
/ (statisk host)
├─ index.html     # huvudsida, allt-i-ett UI
├─ compare.html   # jämför flera målvakter
└─ goalie.html    # detaljsida för en målvakt
```

- Ingen bundler eller build krävs. All logik ligger i sidorna.
- Tailwind via CDN, Chart.js via CDN, Supabase JS v2 via CDN.

Databas & RLS (Supabase / Postgres)

Tabeller

```sql
-- Teams
create table if not exists public.teams (
  id uuid primary key default gen_random_uuid(),
  name text not null,
  owner_id uuid not null references auth.users(id),
  created_at timestamptz default now()
);
```

```sql
-- Medlemskap
create table if not exists public.team_members (
  team_id uuid not null references public.teams(id) on delete cascade,
  user_id uuid not null references auth.users(id) on delete cascade,
  role text not null default 'member',
  created_at timestamptz default now(),
  primary key (team_id, user_id)
);

-- Inbjudningar (valfritt men stöds i UI)
create table if not exists public.team_invites (
  id uuid primary key default gen_random_uuid(),
  team_id uuid not null references public.teams(id) on delete cascade,
  email text not null,
  status text not null default 'pending',
  created_at timestamptz default now()
);

-- Målvakter
create table if not exists public.goalies (
  id uuid primary key default gen_random_uuid(),
  team_id uuid not null references public.teams(id) on delete cascade,
  code text not null,
  name text not null,
  created_at timestamptz default now(),
  unique (team_id, code)
);

-- Matcher
create table if not exists public.matches (
  id uuid primary key default gen_random_uuid(),
  team_id uuid not null references public.teams(id) on delete cascade,
  date date not null,
  opponent text not null default '',
  result text not null default 'V', -- V/O/F
  notes text,
  created_at timestamptz default now()
);

-- Manuell statistik per match & målvakt (minuter/saves/skott)
create table if not exists public.match_goalie_stats (
  id uuid primary key default gen_random_uuid(),
  team_id uuid not null references public.teams(id) on delete cascade,
  match_id uuid not null references public.matches(id) on delete cascade,
  goalie_id uuid not null references public.goalies(id) on delete cascade,
  minutes int not null default 0,
  saves int not null default 0,
  shots int not null default 0,
  created_at timestamptz default now(),
  unique (match_id, goalie_id)
);
```

```
-- Skott-event (x,y i [0..1])
create table if not exists public.shots (
  id uuid primary key default gen_random_uuid(),
  team_id uuid not null references public.teams(id) on delete cascade,
  match_id uuid not null references public.matches(id) on delete cascade,
  goalie_id uuid not null references public.goalies(id) on delete cascade,
  x float8 not null,
  y float8 not null,
  period int not null default 1,
  time_sec int not null default 0,
  result text not null default 'save' -- save|goal|miss|block
);

-- Index som brukar hjälpa
create index if not exists idx_goalies_team on public.goalies(team_id);
create index if not exists idx_matches_team_date on public.matches(team_id,
date desc);
create index if not exists idx_mgs_match on
public.match_goalie_stats(match_id);
create index if not exists idx_shots_match on public.shots(match_id);
```

Hjälpfunktion + RLS-policies

```
-- Är inloggad användare medlem i teamet?
create or replace function public.is_team_member(tid uuid)
returns boolean language sql stable as $$
  select exists (
    select 1 from public.team_members tm
    where tm.team_id = tid and tm.user_id = auth.uid()
  );
$$;

alter table public.teams enable row level security;
alter table public.team_members enable row level security;
alter table public.team_invites enable row level security;
alter table public.goalies enable row level security;
alter table public.matches enable row level security;
alter table public.match_goalie_stats enable row level security;
alter table public.shots enable row level security;

-- Baspolicy: team-medlem får allt på sitt team
create or replace policy teams_all
on public.teams for all to authenticated
using (auth.uid() = owner_id) with check (auth.uid() = owner_id);

create or replace policy team_members_all
on public.team_members for all to authenticated
using (public.is_team_member(team_id)) with check
(public.is_team_member(team_id));
```

```sql
create or replace policy team_invites_all
on public.team_invites for all to authenticated
using (public.is_team_member(team_id) or email = auth.jwt() ->> 'email')
with check (public.is_team_member(team_id));

create or replace policy goalies_all
on public.goalies for all to authenticated
using (public.is_team_member(team_id)) with check
(public.is_team_member(team_id));

create or replace policy matches_all
on public.matches for all to authenticated
using (public.is_team_member(team_id)) with check
(public.is_team_member(team_id));

create or replace policy mgs_all
on public.match_goalie_stats for all to authenticated
using (public.is_team_member(team_id)) with check
(public.is_team_member(team_id));

create or replace policy shots_all
on public.shots for all to authenticated
using (public.is_team_member(team_id)) with check
(public.is_team_member(team_id));
```

> **Obs:** Anpassa `teams_all` om du vill att fler än ägaren ska se själva team-raden. Appens övriga läsningar går via `team_members`, `goalies`, `matches` osv, så normal drift funkar även med strikt team-policy.

Hur appen funkar (översikt)

Autentisering

- Email/Password via Supabase Auth. Session hålls i LocalStorage och auto-refreshas.
- `boot()` läser sessionen och visar antingen **Auth-formulär** eller **appen**.

Team

- Vid första start skapas (om behövs) ett standardteam och du läggs som `owner`.
- Du kan skapa fler team, byta team via dropdown och bjuda in användare med e-post.
- Inbjudningar lagras i `team_invites` och kan accepteras i UI.

Målvakter

- Varje team har sina egna målvakter i `goalies` (unik `code` per team).
- `index.html` seed:ar några standardkoder (t.ex. LL, IE, EH, LK) om de saknas.

Matchflöde

1. Fyll datum, motstånd, resultat och ev. notering.
2. Varje målvaktsrad har tre fält: **Min**, **Räddn**, **Skott**.
3. Knappen **+ Skott** öppnar en modal med **rink** där du klickar för att lägga skott-event. Dessa buffras i minnet tills du sparar matchen.
4. **Lägg till match** skapar matchen, sparar manuell statistik (om ifylld) och **alltid** sparar skott-eventen till `shots`.

Matchlista & statistik

- För varje match visas rader per målvakt med **Min** (från `match_goalie_stats`), **Räddn** & **Skott** där källan prioriteras enligt: 1) Finns **events i** `shots`? Använd dessa (räddn = antal `save`, skott = `save + goal`). 2) Annars använd **manuella fält** (`saves` / `shots`).
- En **Summa**-rad summerar Min, Räddn, Skott och visar antal mål i parentes.
- Varje match har en **Skottkarta** (toggle) med filter per målvakt + legend (Räddn/Mål/Miss/Block).
- **Ta bort** raderar matchen (och associerade shots via FK `on delete cascade`).

Grafer

- `index.html` – **Räddd% över tid** för vald målvakt (inkl. 3-matchers glidande medelvärde).
- `compare.html` – överlagrad **Räddd%** för flera målvakter och **Skott & Räddningar per match (summa per datum)**.
- `goalie.html` – målvaktens egen **Räddd% över tid** + **Skott/Räddningar per match**.

Export / Import

- **Export JSON** – hela teamets målvakter + matcher (för backup).
- **Export CSV** – semikolon; rubriker `CODE_minutes/saves/shots` per målvakt.
- **Import JSON/CSV** – upsert på match (nyckel: `date + opponent`), ersätter `match_goalie_stats` för matchen.

Koordinatsystem för skott

- `x`, `y` i intervallet **[0..1]**, med **(0,0)** uppe till vänster i rinkens SVG.
- I UI ritas mål som röd **X**, räddning grön **fylld** cirkel, miss grå ring, block blå kvadrat.

Vanliga problem & felsökning

1) Kan inte logga in - Kontrollera `SUPABASE_URL` & `SUPABASE_ANON`. - Email/Password aktiverat i *Auth → Providers*. - Din utvecklings-/produktions-domän är tillagd i *Auth → URL Configuration → Redirect URLs*.

2) Skott sparas inte - RLS/policy kan stoppa INSERT. Appen visar felmeddelande från PostgREST om så sker. - Säkerställ att `shots_all` policy **med** `with check` finns och att `public.is_team_member(team_id)` returnerar true för din användare.

3) Matchlista visar inga skott - Om endast manuella fält används, visas de (Räddn/Skott) – annars dyker event upp när `shots` är sparade. - Se att `refreshAll()` laddar **både** `shotStatsByMatch` och `manualStatsByMatch` och skickar in båda till `renderMatchList(...)`.

4) ".group is not a function" - Supabase JS v2 stödjer inte `.group()` på klienten. Appen hämtar rader och grupperar i JS – se att din version matchar.

5) "...get is not a function" i `renderMatchList` - Uppstår om man råkar skicka ett objekt där koden förväntar sig en `Map` (eller tvärtom). I den här koden är funktionen robust och tar `{ shotStatsByMatch, manualStatsByMatch }` (Maps) korrekt.

6) "Ta bort" gör inget - Sidan använder **event-delegation**. Kontrollera att `renderMatchList` bara kopplar handlern **en gång** (`__hasDeleteHandler__`).

Deploy

- Hostas som **statisk sajt** (Netlify, Vercel, Cloudflare Pages, GitHub Pages). Ingen server behövs.
- Lägg dina verkliga `SUPABASE_URL` och `SUPABASE_ANON` i filerna innan deploy.
- Glöm inte att lägga till produktionsdomänen i Supabase **Redirect URLs**.

Anpassningar & tips

- **Fler eventfält**: Utöka `shots` (t.ex. skottzon, avslutstyp, xG) och räkna i klienten.
- **Periodfilter i skottkarta**: Lägg ett select (1/2/3/OT) i panelen och filtrera `shots` innan render.
- **Automatisk målvakt från kod**: Appen skapar saknade målvakter vid behov när man sparar manuella stats.
- **Ytterligare roller**: Utöka `team_members.role` och policies därefter (t.ex. `coach`, `analyst`).
- **OAuth**: Lägg till Google/Apple-login i Auth Providers om ni vill.

Roadmap / Idéer

- Skottkarta: värmekarta, puckspår, tidsordning (1..N) på målmarkörer.
- Avancerad jämförelse: normalisering per motståndsnivå, 5-matchers rullande fönster.
- Export av skott-events (CSV/JSON) och import av samma.
- Offline-läge (IndexedDB) med sync mot Supabase.

Licens

Detta är ett hobby/projektupplägg. Använd, forka och anpassa fritt.