

Nedenfor har jeg gjort **strukturet research** på hvordan etablerte friidretts-statistikksystemer er bygget (TFRRS, Tilastopaja, Friidrottsstatistik.se, World Athletics, Athletic.net), og deretter **oversatt dette til en konkret spesifikasjon** for et norsk, skalerbart resultat- og statistikksystem som håndterer ~72 000 resultater/år og full klikkbar navigasjon mellom alle objekter.

Jeg deler svaret i 5 deler:

1. Hva de beste systemene faktisk gjør (research)

◆ TFRRS

Styrker

- Relasjonell kjerne: Resultat → Utøver → Skole/Klubb → Stevne → Sesong
- Alle sider er *filtrerte visninger av samme database*
- Tydelig URL-struktur:
 - /athletes/{id}
 - /meets/{id}
 - /results/{id}
- Resultater er **atomiske** (én prestasjon = én rad)

Svakheter

- Lite fleksibel visning
- Begrenset historikk før college
- Ingen sanntidsoppdatering

👉 **Lærdom:** Sterk relasjonsmodell, enkel navigasjon, konsekvent URL-design.

◆ Tilastopaja

Styrker

- Tett kobling mellom:
 - Utøver ↔ Øvelse ↔ Sesong ↔ Stevne
- Veldig god historikk (hele karrierer)
- «Sesongkort» per utøver
- Resultater normalisert på:
 - Vind
 - Innendørs/utendørs
 - Aldersklasse

Svakheter

- Lukket system

- Litt treg navigasjon

👉 **Lærdom:** Sesong- og øvelsesfokustert navigasjon er kritisk.

◆ Friidrottsstatistik.se

Styrker

- Ekstremt konsekvent:
 - Alle sider = filtre
- Samme visning kan brukes til:
 - Årsstatistikk
 - All-time
 - Aldersklasser
- Veldig effektiv for statistikknerder

Svakheter

- Lite moderne UX
- Lite sanntid

👉 **Lærdom:** *En sann datamodell, mange visninger.*

◆ World Athletics

Styrker

- Absolutt referanse for:
 - Ranking
 - Rekorder
- Sterk autoritetsmodell
- Gode koblinger:
 - Resultat → Ranking → Biografi → Mesterskap

Svakheter

- Ikke nasjonalt detaljnivå
- Begrenset klubbinformasjon

👉 **Lærdom:** Skill tydelig mellom *resultatdata* og *aggregerte visninger* (ranking, rekorder).

◆ Athletic.net

Styrker

- Nær sanntid
- Mange datakilder
- Ulike formater
- Brukergenerert input
- Sterk stevne-logikk

Svakheter

- Datakvalitet varierer
- Duplikater

👉 **Lærdom:** Du trenger robust import- og valideringspipeline.

2. Grunnprinsipp: Én database – mange innganger

Alle systemene over følger i praksis samme filosofi:

Alt er samme data – bare sett gjennom forskjellige linser

Det betyr:

Objekt	Eksempel
Utøver	Person + historikk
Resultat	Én prestasjon
Stevne	Samling av resultater
Øvelse	100 m, stav, lengde
Sesong	2025
Klasse	Senior, U20, U18

3. Foreslått kjerne-datamodell (Norge)

🌿 Kjerne-entiteter (normalisert)

```

ATHLETE
- athlete_id (PK)
- name
- gender
- birth_date
- nationality
- club_id
- active_from / active_to

```

```

CLUB
- club_id
- name
- federation
- location

```

EVENT

- event_id
- discipline (100m, PV, LJ)
- indoor_outdoor
- gender
- implement (høyde, hekk, vekt)

MEET

- meet_id
- name
- location
- start_date
- end_date
- organizer
- level (lokalt, nasjonalt, internasjonalt)

RESULT

- result_id
- athlete_id
- event_id
- meet_id
- date
- performance
- wind
- place
- round (heat, final)
- status (OK, DNS, DQ)
- record_flags (NR, PB, SB)

👉 **Resultat er den absolutte atom-enheten**

4. Navigasjon (det viktigste)

🔄 **All navigasjon må være to-veis**

Eksempel: Klikkbar flyt

```
Årsstatistikk 2025 (stav menn)
  → Utøver (Sondre Guttormsen)
    → Karrierestatistikk
      → Stevne (NM innendørs 2025)
        → Resultatliste
          → Andre utøvere
```

📌 **Alle disse er bare filtre:**

Side	Filter
Årsstatistikk	season + event
Utøverside	athlete_id
Stevneside	meet_id
Resultatliste	meet_id + event

5. Import, oppdatering og skala (kritisk for 72 000+ resultater)

Resultat-import (må designes eksplisitt)

Støttede formater

- CSV
- Excel
- XML
- Live feed (JSON)
- Manuell registrering

Pipeline

RAW FILE
→ Parsing
→ Normalisering
→ Validering
→ Deduplicering
→ Publisering

Datakvalitet

- Tidsstempel på alle importter
- Versjonering av resultat
- Original kilde lagret
- Mulighet for korrigering

6. Hvordan dette spesifiseres (kort kravliste)

Funksjonelle krav

- Års- og historisk statistikk
- Full klikkbar navigasjon
- Sanntids oppdatering av stevner
- Utøversider (som minfriidrettsstatistikk / Tilastopaja)
- All-time lister
- Aldersklasse-filtre
- Innendørs/utendørs
- Vindjustering

Arkitektur

- Relasjonell DB (Postgres)
 - API-first (REST / GraphQL)
 - Frontend = filtrerte views
 - Caching av populære lister
-

7. Viktig strategisk anbefaling

! Ikke bygg «sider» – bygg **visninger av samme data**

Alle systemene som har overlevd i 20+ år har gjort nettopp dette.

Neste steg (kan gjøre sammen)

1. Lage **ER-diagram**
2. Definere **API-enderpunkter**
3. Lage **URL-struktur**
4. Skissere **frontend-navigasjon**
5. Lage **import-spesifikasjon**

Si hva du vil gå videre med 🙋