Engineer Advisory

Built a loyalty wallet, leagues and badges

Built the loyalty and reputation system — 67 functions over a 31-table ledger, with leagues, badges and a redemption shop — taking a row lock on the balance to close the double-spend window.

2025

67

functions over a 31-table loyalty ledger

Situation. Professional networking has a cold start problem: the platform is worth using once other people are already using it, and until then there is little reason to come back. The usual lever is a rewards system — points for contributing, standing that reflects reputation — which is easy to describe and treacherous to build, because the moment points can be spent they are money, and every mistake money makes is available to be made here.

Task. Contribution had to be measurable and rewardable, with a balance that could not be spent twice.

Action. The loyalty schema runs to 31 tables and 67 functions, with standing in a further 5 tables and 32 functions, and discovery and personalization schemas alongside them to decide what a given member sees. On top of the ledger sit leagues, badges and a redemption shop where a balance turns into something real. The part that took the care is the oldest bug in the book: check the balance, then spend it, and two requests arriving together both pass the check. Every mutation takes a row lock on the wallet before reading it, so the second request waits for the first to finish rather than racing it. That the wallet is in the same database as everything else is what makes it possible at all — the balance and the thing it bought commit together or not at all.

Result. Contribution is measured and rewarded, and a balance is arithmetic rather than an approximation. Row locking is the slower answer and was chosen anyway: contention on a wallet means members wait their turn, and the

alternative is a member spending the same points twice and someone reconciling it by hand afterwards.

Bring this experience to your project

Use “Built a loyalty wallet, leagues and badges” as a starting point for a similar challenge in your organisation. The services behind this work are available for your project. Turning what the business wants into something a developer can build, and saying out loud what is not worth building.

Explore our services Get in touch →

Look behind the result

Read the situation, decisions and result in this example. We can discuss which parts transfer to your constraints and which need another approach. Built a loyalty wallet, leagues and badges — 67 functions over a 31-table loyalty ledger

Explore our services Get in touch →

Want to reward the people who show up?

Talk through your project with the engineer responsible for delivery. We agree the scope, price basis and handover in writing before work begins.

Role context: Database Engineer. It draws on Backend Engineering, Databases, PostgreSQL, SQL, Performance Tuning and Product & Requirements. And while you are here, we are rather good at Backend & API Development, Database Design & Modeling and Product Strategy & Requirements.

Get in touch Explore our services

One piece of **a longer record** — and it is all on this site.

Every entry is written the same way: the situation, the task, what we did and what changed.

See the whole record

Define the improvement you need

Bring the decision, the people affected and the options already considered. We can make the tradeoffs explicit and leave a documented recommendation.

See what it costs Get in touch →

Want to reward the people who show up?

Ask about the decisions behind “Built a loyalty wallet, leagues and badges”. You will speak with the people who can explain the work and assess your own project.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://advisory.engineer.company/portfolio/built-a-loyalty-wallet-leagues-and-badges/