

Finance Manager Review

Key Findings

- [High] Returning the JPA entity from the account endpoints (AccountController.java:34, AccountController.java:49)
- [High] The project includes Spring Security (pom.xml:33-76) but defines no security configuration. Without security configuration, the application is vulnerable to various attacks.
- [Medium] Database credentials (username/password kevin1995) ship in source control in application.properties (application-dev.properties:3, application-dev.properties:4)
- [Medium] AccountService.createForUser trusts the request body (AccountService.java:31-36). If starved of input, the application may crash or become unresponsive.
- [Low] application-dev.yml:7 targets port 5432 while application-dev.properties:3 targets 5433, making it difficult to run multiple instances simultaneously.
- [Low] No automated tests cover registration or account flows (services or controllers). Add happy-path tests to ensure the system behaves as expected under normal conditions.

Open Questions

- Are you planning to introduce JWT authentication soon? Knowing that timeline will guide whether to implement it now or defer it to a later phase.
- Should the account API ever expose user context, or will a DTO that hides internal IDs be the long-term solution?

Project Overview

Backend Spring Boot service for personal finance: users register, create accounts, and list accounts across multiple categories.

Next Steps

1. Introduce DTOs plus validation to stop leaking internal fields and to enforce required payloads.
2. Add a security configuration (or disable the starter) so documented endpoints are reachable, then consider integrating OAuth2.
3. Externalise secrets and pick a single dev config path to simplify setup.