- So services can be used as repositories? In theory, no, but in practice, yes. And a lot of developers use services instead of repositories.
- Can services be used together with repositories? Yes, but I don't recommend this approach.

When you have repositories and services as well in a project, this is the main idea:

- Each model has a repository.
- Repositories contain database queries.
- When needed, you create a service to a model (or a set of models, just as we discussed in the repository chapter). For example, the Todo class has more complex notification logic, so you might add a `TodoNotificationService`.

This approach is fine and can work, but here's the main problem:

- You end up with inconsistent classes. When you're working with the `Todo` model, you have a repository and a service. But when you're working on a `Project` related feature, you only have a repository because the project doesn't require a service class. And in the case of `Todo`, you won't be able to tell quickly if a method lives in a service or a repository.
- So, in general, your features are spread across services and repositories and a bit harder to reason about.

This isn't very objective, so it may work very well for you! As for me, I stick with actions.