Why E-mail Marketing?

As you can see from these chapters, building e-mail marketing software can be pretty complicated! This is why I chose this topic:

- The domain model is perfect for introducing DDD. From these features, we can already identify some important domains: subscriber, broadcast, sequence, and report. Probably all of these can be a separate domain.
- The business logic is complicated. Just look at the figure above. Or think about sending 10000 e-mails or importing 50000 subscribers. We will face some challenges.
- We need to avoid code duplication. A broadcast and a sequence e-mail sound very similar to each other. We probably need some abstraction.
- Reports everywhere! Writing reports-related code can be pretty fun but often become a big mess.

It's interesting from the frontend perspective as well:

- I will use Inertiajs because it's a handy tool. If you're not familiar with it, don't panic. From the BE perspective, it's only one line of code in the controller.
- But we also need some API endpoints to track the open and click rates and create subscribers. These requests come outside of our application (from an embedded form or a script in the e-mail's content).
- Because of this, technically, we have three applications:
    - Web. We are building with Inertia, used by logged-in users.
    - API. We are building with Laravel, used by external parties.
    - Console. Scheduling sequences or importing subscribers.

For these reasons, I think ConvertKit is the perfect example to replicate and learn Domain-Driven Design!