FLEX - Erlbank

Simon Härer

Created: 2021-01-20 Wed 08:31

Introduction "Erlbank"

A simple bank with accounts, transactions & bank statements.

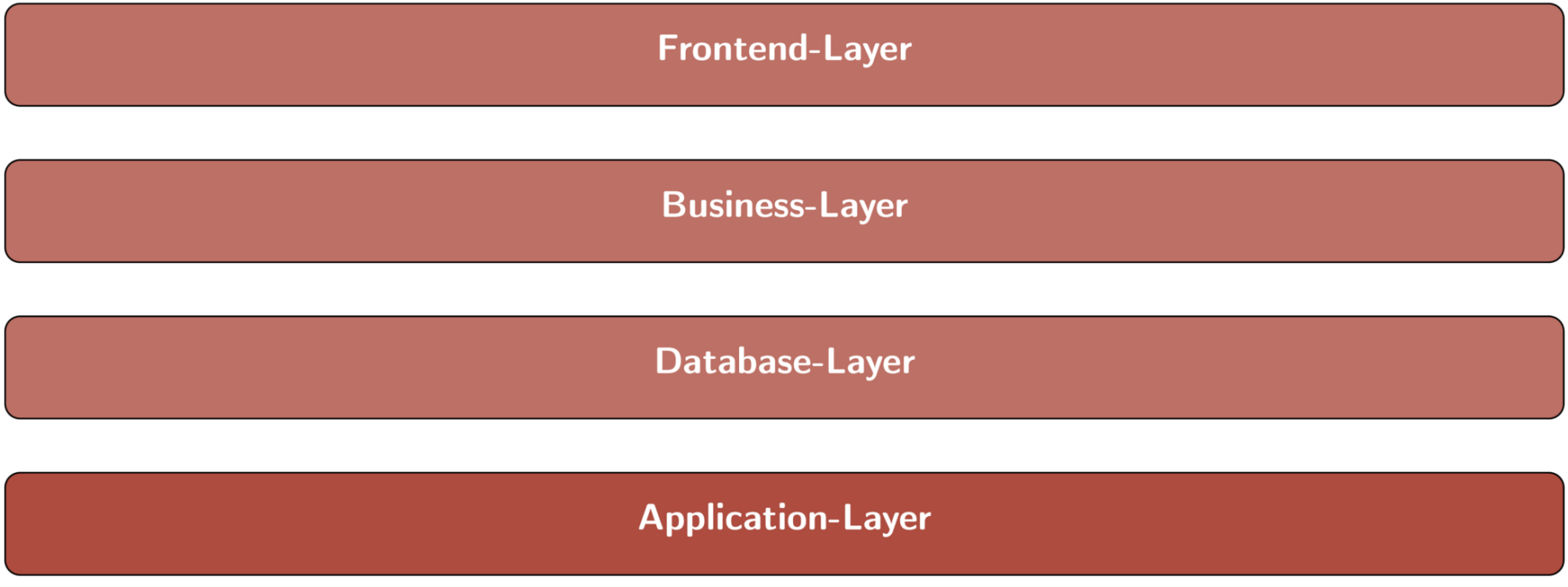
- Typical monolithic horizontal architecture
- Divided into standard layers:
  - application layer
  - business layer
  - data layer
  - front-end layer

Erlbank

Erlbank is ...

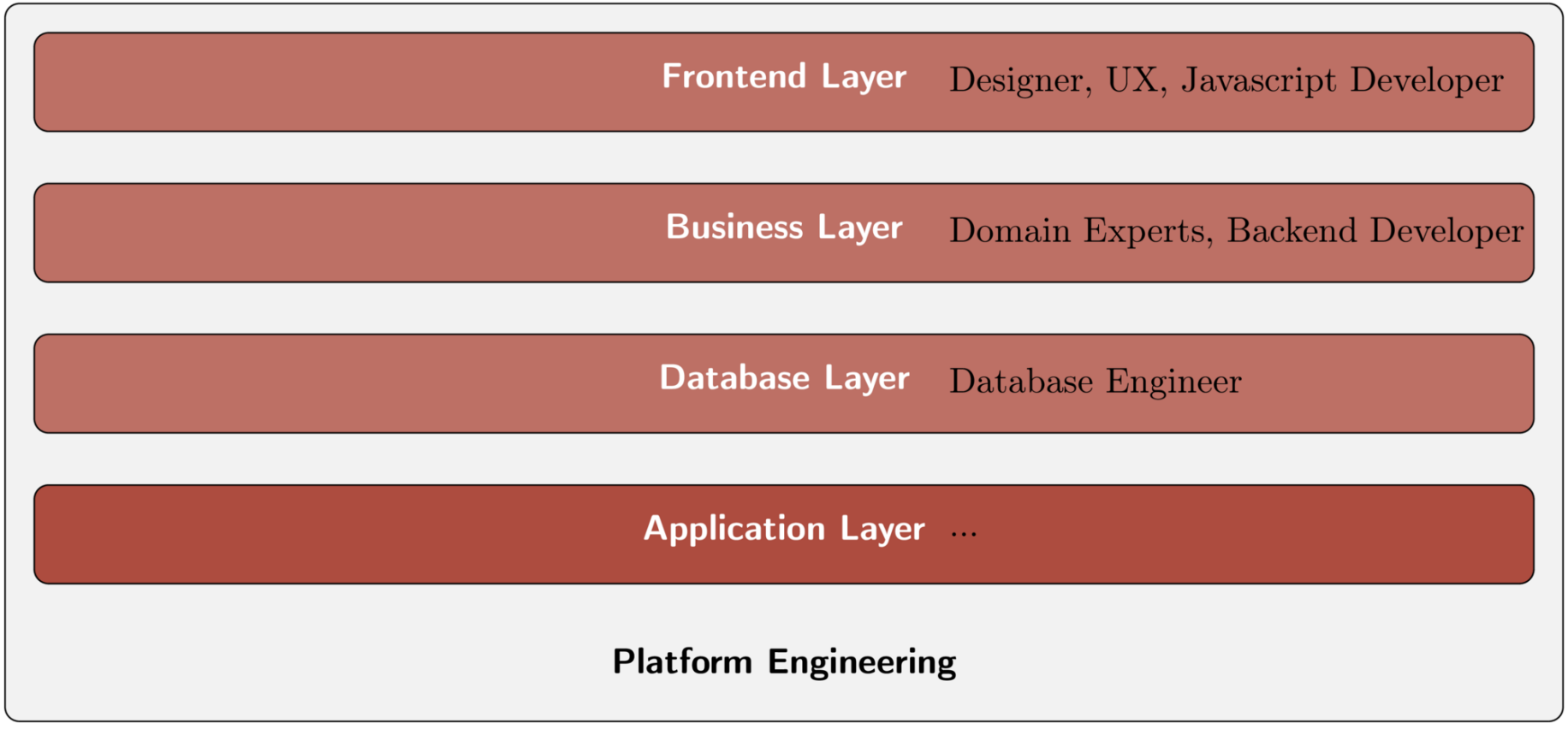
- a monolithic banking backend
- a classical 3-layer architecture, consisting of a database, business and frontend layer
- an educational banking backend, featuring account creation, transactions and bank statements

Architecture Stack



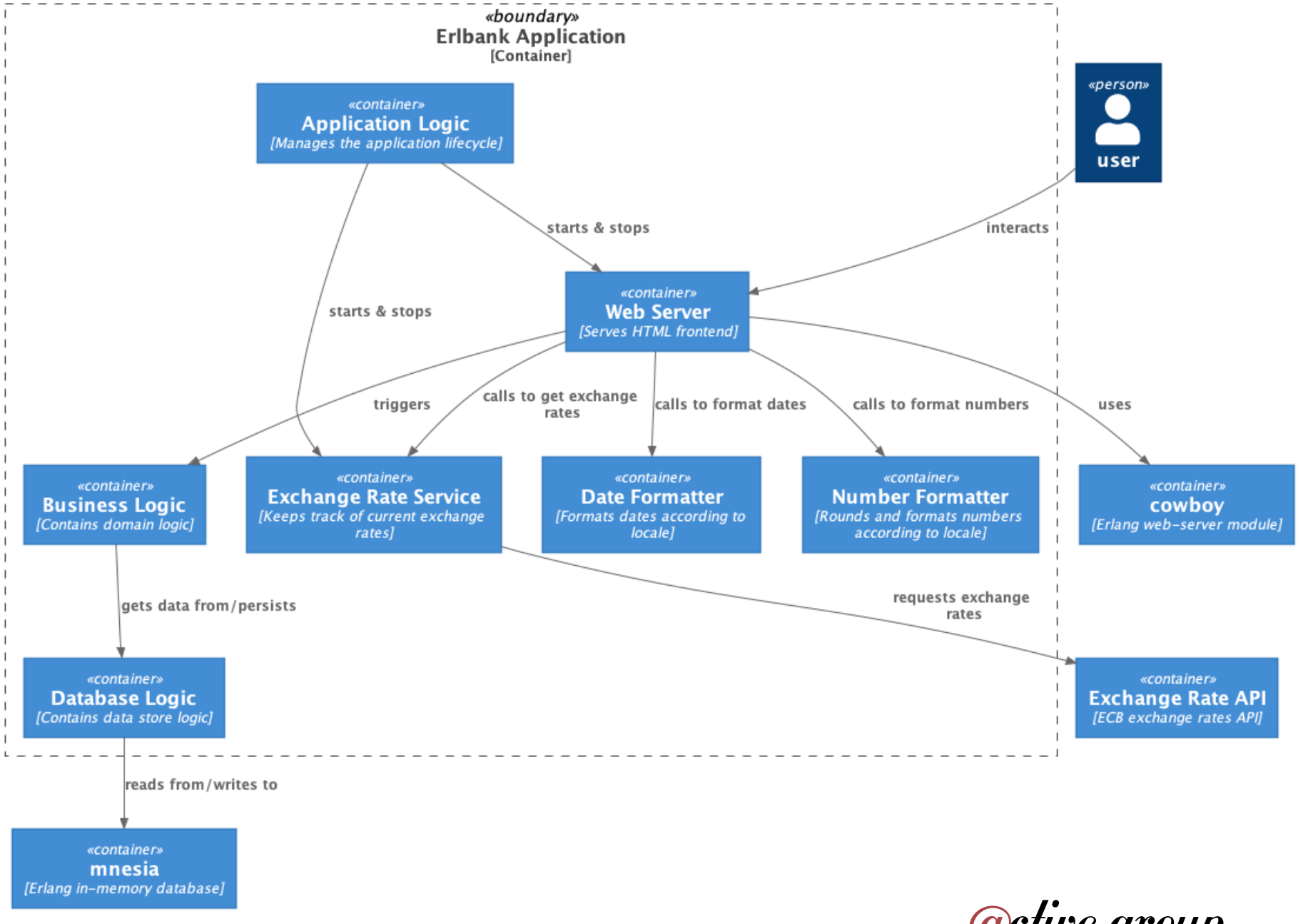
@ctive group

Teams



@ctive group

Component Diagram


«boundary»
Erlbank Application
[Container]
«container»
Application Logic
[Manages the application lifecycle]
«person»
user
starts & stops
interacts
starts & stops
«container»
Web Server
[Serves HTML frontend]
triggers
calls to get exchange rates
calls to format dates
calls to format numbers
uses
«container»
Business Logic
[Contains domain logic]
«container»
Exchange Rate Service
[Keeps track of current exchange rates]
«container»
Date Formatter
[Formats dates according to locale]
«container»
Number Formatter
[Rounds and formats numbers according to locale]
«container»
cowboy
[Erlang web-server module]
gets data from/persists
requests exchange rates
«container»
Database Logic
[Contains data store logic]
«container»
Exchange Rate API
[ECB exchange rates API]
reads from/writes to
«container»
mnesia
[Erlang in-memory database]

@ctive group

Exercise: Analyze Erlbank's Architecture

Which problems can occur if the app grows? Discuss for each of the following levels:

- Development
- Technology
- Organization
- Quality

Answer:

- Problems depend on the quality goals and challenges the architecture is facing
- Do we need scaling? Fast time to market?
- Analysis in following slides

Problems in Development with Layer-Based Teams

- Introduction of new features requires multiple teams with different technological focus to collaborate. This creates dependencies between teams and hampers parallel development.
- Every change in code could have influence on other parts that should stay untouched
- Tests need to be done on the whole code base. Thus, testing takes very long.
- Dependencies prevent independent deployment.

Development and deployment is slow.

Technological Problems

- Parts of the app can tear down the whole rest leaving everything dysfunctional
- Scaling the app is very hard
- Replacing technology may affect every team

Scaling and refactoring is hard.

Organizational Problems

- New features must be coordinated between teams with own KPIs.
- Process is very inflexible, since there are many dependencies in implementation.
- Based on Conway's law, the organization may not be separated into logical domain-driven units, but there is a huge IT department.

Flexibility and domain scaling is expensive.

Organizational Problems – Conway's Law

“Organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations.”

Melvin E. Conway: How Do Committees Invent?

Implications of Architectural Problems

- These problems cost money.
- Scaling application development is hard.
- If the app is not flexible enough to adapt easily to new scenarios, it could threaten the company's existence.
- Long time to market for new features.

Architectural Drivers

- Coupling across organizational boundaries create organizational depencies.
- Communication overhead and cost is significant across organizational boundaries.
- Organizational dependencies hamper scaling the development team.
- Organizational dependencies hamper adaptation and innovation.
- Organizational dependencies prolong time-to-market.

Exercise: Propose Solutions for Erlbank Architectural Problems

- Formulate quality goals based on the analyzed problems
- How can we restructure to achieve the quality goals?
- How can you achieve fast deployment and thus fast feedback?

Answer:

- Quality goals can include: Robustness, fast time to market, independent development, better testing, ...
- It depends on the quality goals defined in the first part of the exercise. We will focus on those that arise because of the monolithic architecture. Thus:
  - Break up the monolithic architecture
  - Participants are motivated to propose their own solutions, not necessarily following the course of the training

Discussion: Trade offs for proposed solutions

Every answer to the former exercise has advantages, but comes with drawbacks, too. Discuss what they are and try to judge the long-term consequences.

Code Session – Solve the problems! Take 1.

Participants restructure the architecture in different teams and try to solve the problems in the actual code.

The result depends on the choices the participants make. Participants are explicitly allowed to be creative and make mistakes.

Exercise: Discuss the Result of Code Session 1

- Discuss coupling: How are the different parts of the application coupled?
- Discuss the hard problems the participants were facing, when restructuring based on their first idea
- Are there still problems with the restructured applications?

Answer:

The answer to this question depends on the strategies chosen and implemented by the participants.

Implications of Drivers

- Processes must be – as much as possible – local to organizational units.
- Coupling across organizational boundaries must be minimized.
- Technologies must support separate development and deployment.
- Deployment must be fast and reversible – as supported by the ops infrastructure.