An Enrichment System Integration

Scala in a Fintech Domain

Jonathan Deyrson
Scala Functional Designer
@deyrson

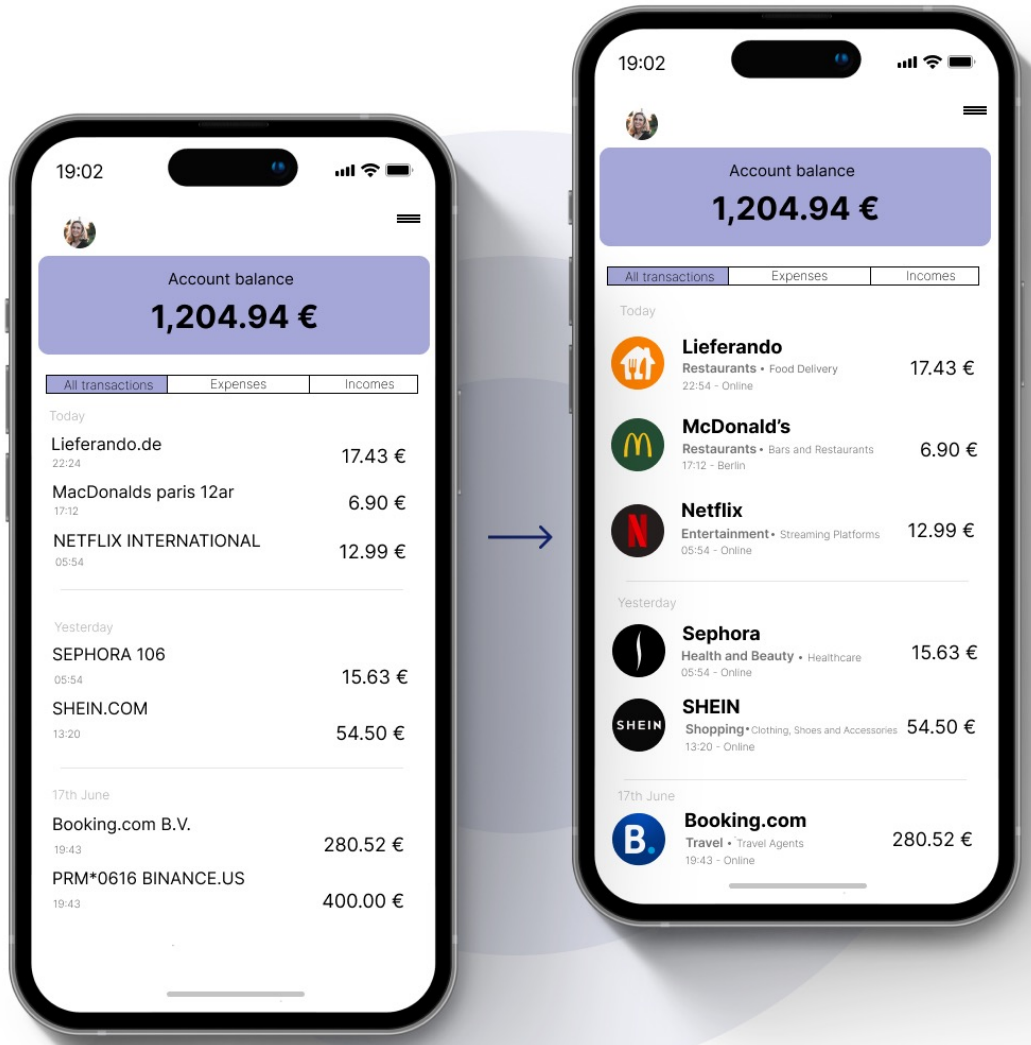
Xebia

First Part

- Use Case (HLD)
- Context Mapping
- Architectural Design

Second Part

- Detailed Design
- Recap
- Scala FP Ecosystem



Data Enrichment as a Product

Raw	sq*Star buc debitcard 20240304 US NY10016	28.99	debit	
Token	sq	*	star	buc	debit	card	20240304	NY10016	US
Predicted	Processor Square	Merchant Starbucks	Transaction type Debit card	Date 20240304	Location NY10016 US				

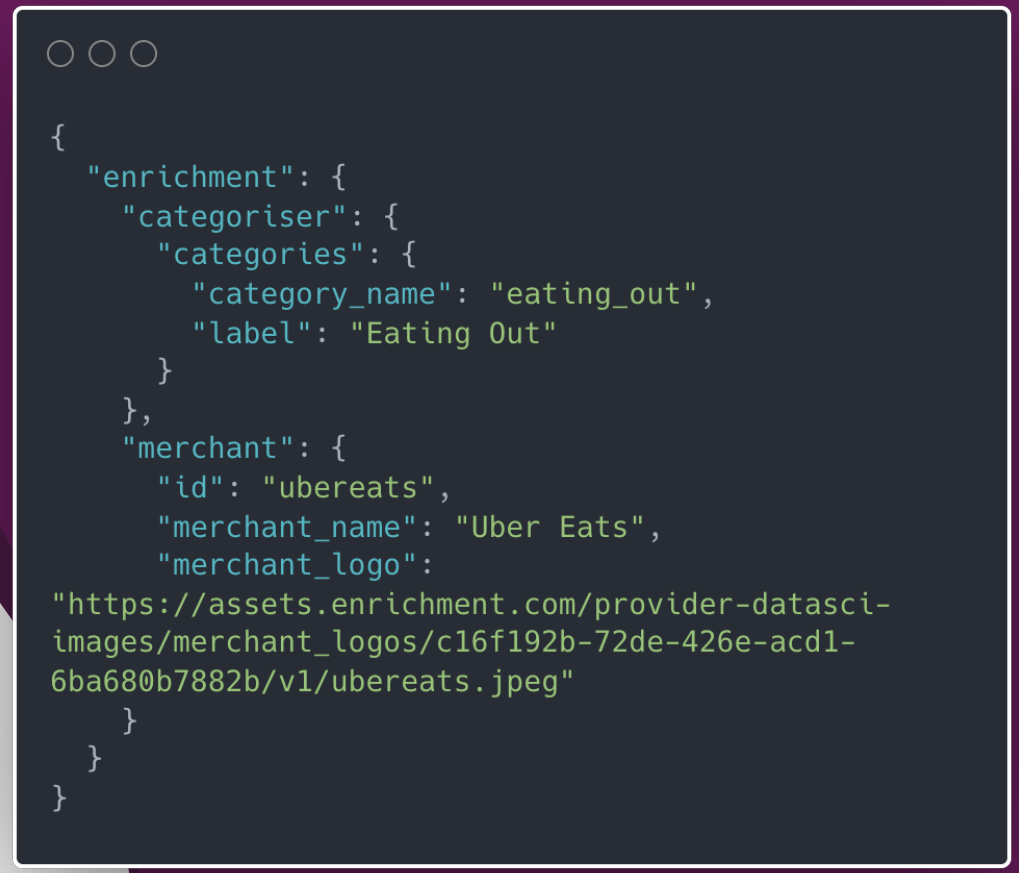
Enrichment Process

raw data
- Financial Transactions

enrichment
- Providers
  - Triple
  - Belvo
  - Plaid

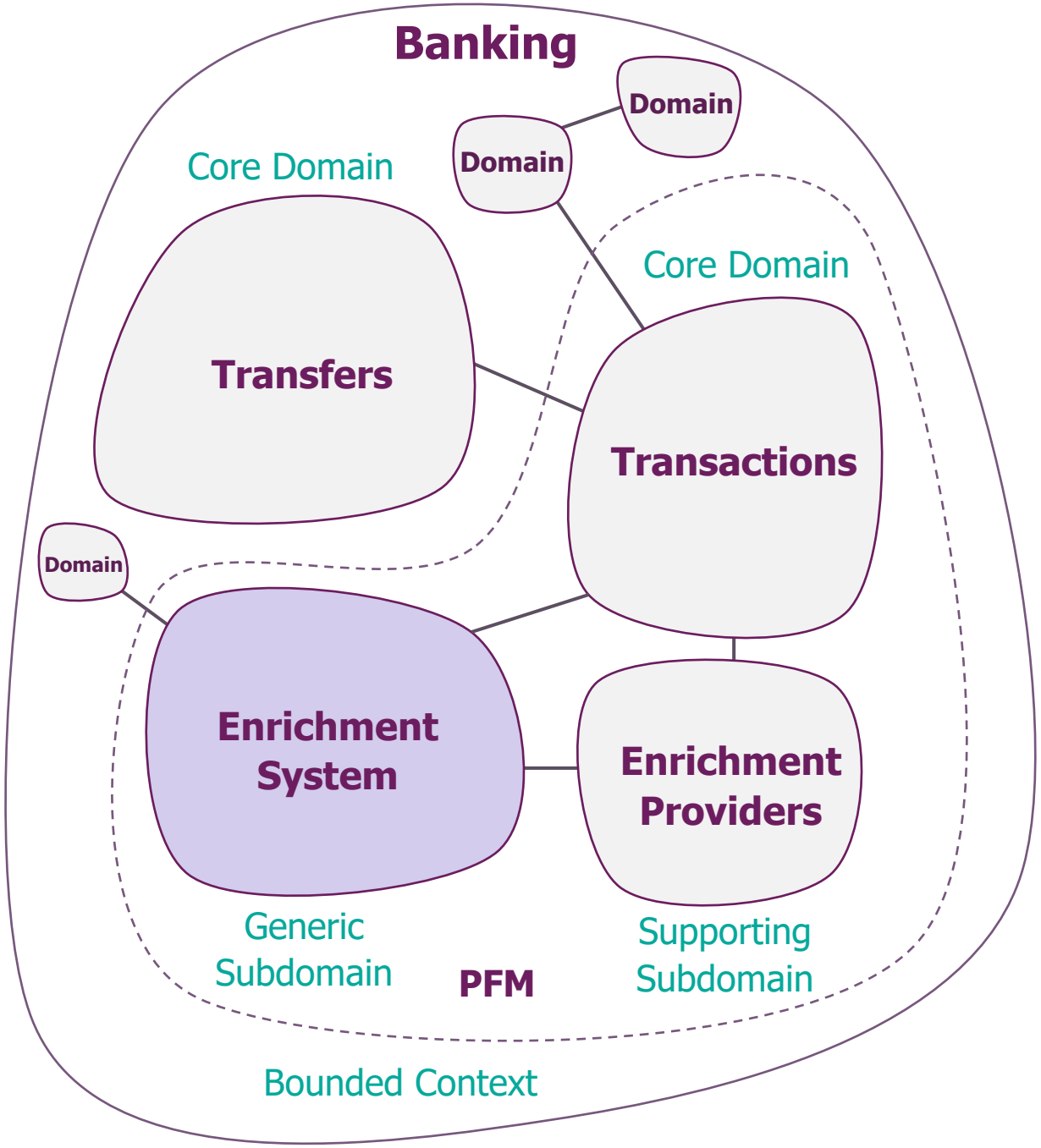
enriched data
- Category
- Merchant
  - Name
  - Logo

```
{
  "raw_transaction": {
    "transaction_id": "e01707b4-e134-562e-8895-eb576883b709",
    "account_id": "5dedaa5a-cb39-406a-a354-d0b8b1219ef8"
  }
}
```



```
{
  "enrichment": {
    "categoriser": {
      "categories": {
        "category_name": "eating_out",
        "label": "Eating Out"
      }
    },
    "merchant": {
      "id": "ubereats",
      "merchant_name": "Uber Eats",
      "merchant_logo":
"https://assets.enrichment.com/provider-datasci-
images/merchant_logos/c16f192b-72de-426e-acd1-
6ba680b7882b/v1/ubereats.jpeg"
    }
  }
}
```

https://jointriple.com/product/transaction-data-enrichment



Domain and Scope

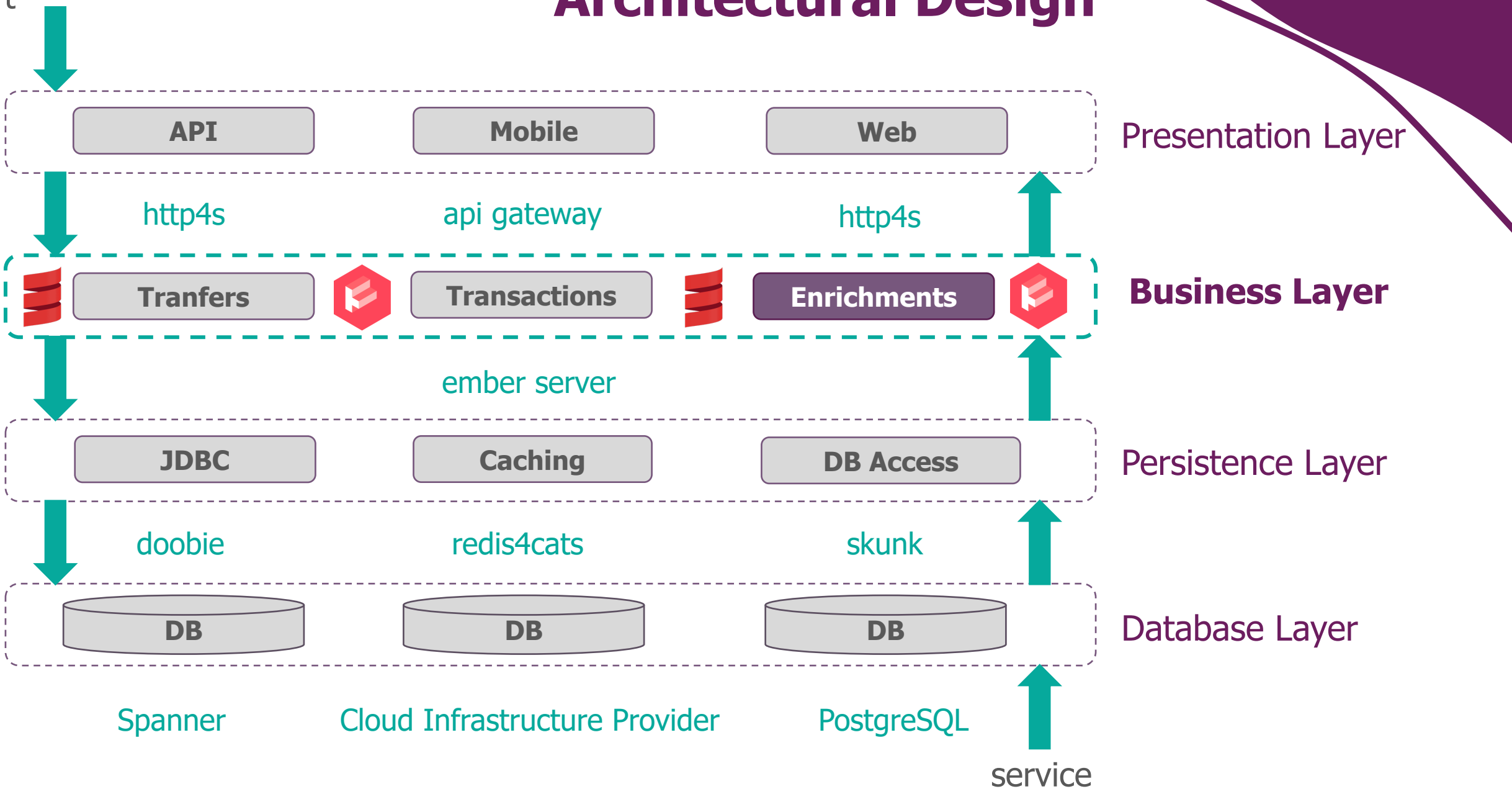
Domain-Driven Design (DDD)

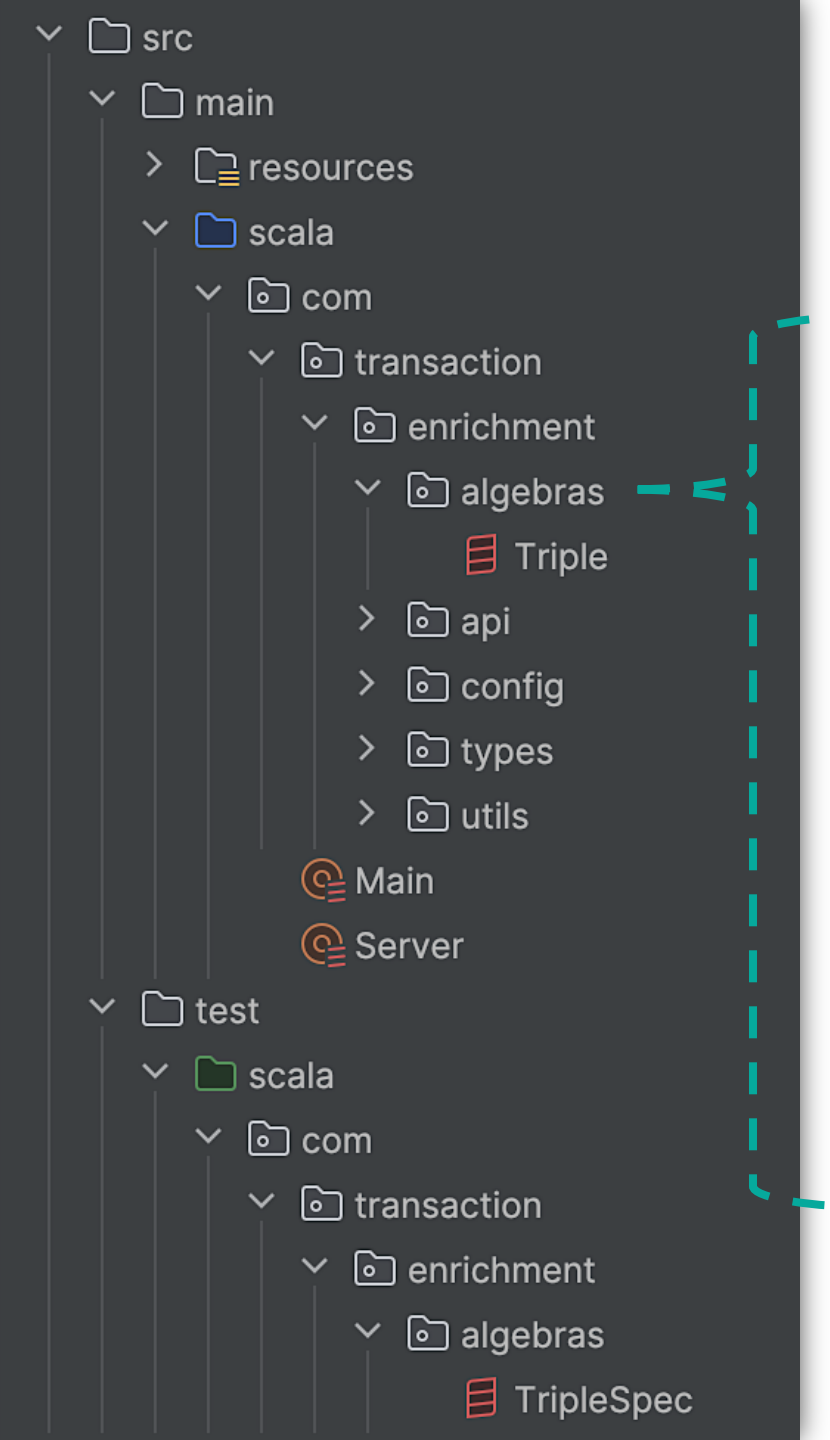
- Strategic Patterns (Architectural Design)
- Tactical Patterns (Detailed Design)

Critical Software Systems

- Safety Critical
- Mission Critical
- **Business Critical**

Architectural Design





business

domain

core

algebras

DSLs

Project Scaffolding

A minimal example structure

http4s giter8 template

This template is fixed on `cats.effect.IO`. For a final tagless version, see http4s.g8.

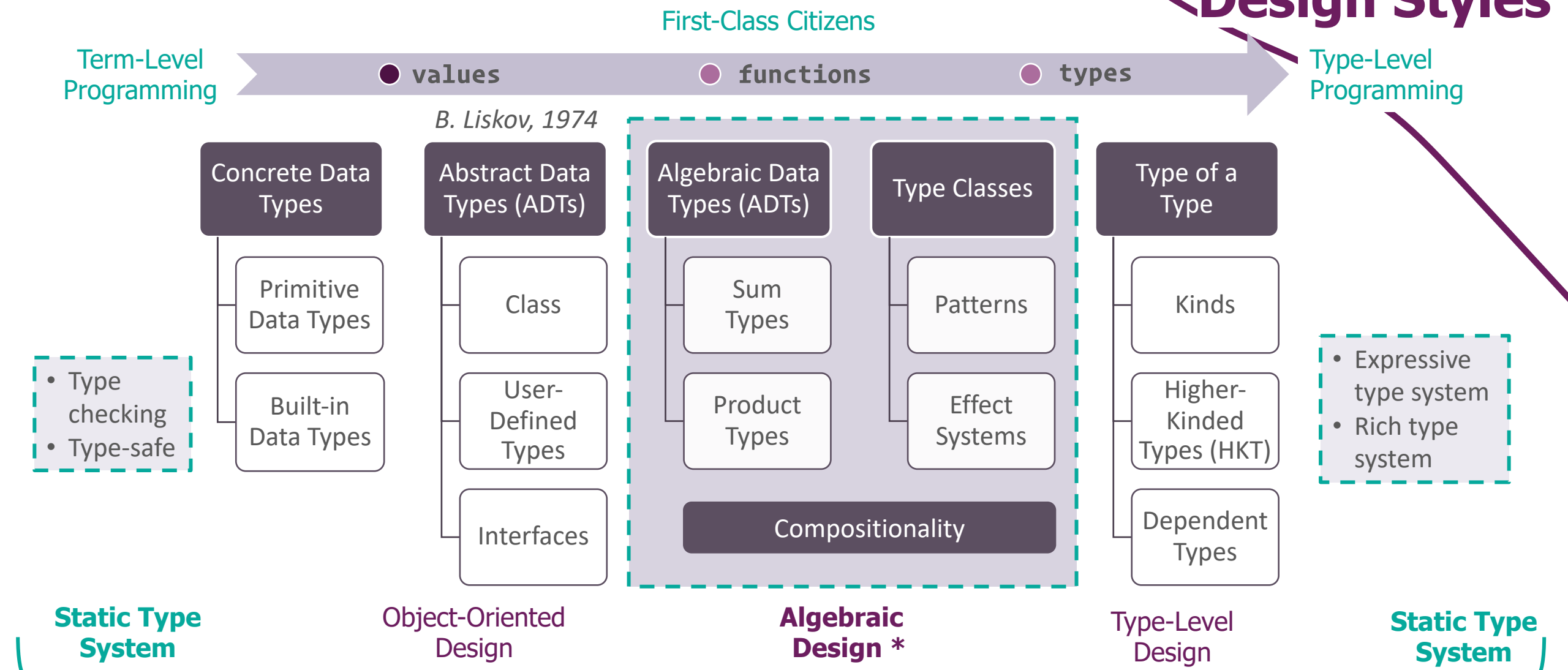
Instructions

Generate an http4s service on the ember backend with Circe.

1. Install sbt
2. Create your project:
   - Scala 2: `sbt new http4s/http4s-io.g8`
   - Scala 3: `sbt new http4s/http4s-io.g8 --branch 0.23-scala3`
   - Java 8: `sbt new http4s/http4s-io.g8 --branch 0.23-java8`
3. `cd quickstart`
4. `sbt run`
5. `curl http://localhost:8080/hello/$USER`
6. Learn more

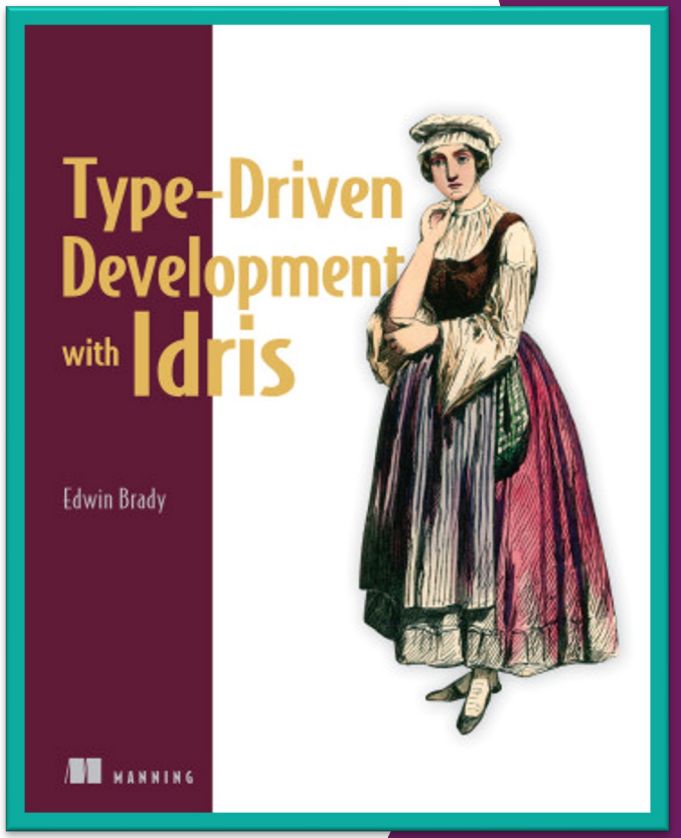
https://www.foundweekends.org/giter8/usage.html

Design Styles



* *Just as **algebra** involves operations that follow well-defined rules (like addition and multiplication), **algebraic design** structures data and computations according to similar rules, making them **mathematically** sound.*

“Types allow you to make models explicit in code and ensure that your implementation of a program matches the model in your head.”
—Edwin Brady, 2017



The lost art of Detailed Design

Caveat

Code as Design: Three Essays by Jack W. Reeves

PDF of All 3 Essays | Discussion Page

The following essays by Jack W. Reeves offer three perspectives on a single theme, namely that programming is fundamentally a design activity and that the only final and true representation of "the design" is the source code itself. This simple assertion gives rise to a rich discussion —one which Reeves explores fully in these three essays.

https://www.developerdotstar.com/mag/articles/reeves_design_main.html

Software Design

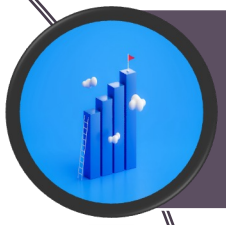
- Architectural Design
- Detailed Design

Big design up front (**BDUF**) is a software development approach in which the program's design is to be completed and perfected before that program's implementation is started. It is often associated with the waterfall model of software development.

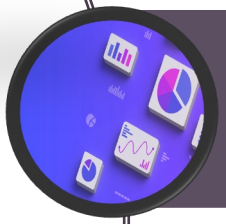
https://en.wikipedia.org/wiki/Big_design_up_front

Minimal Use Case

An Enrichment System Integration



*This use case explores an **approximation** of enriching raw transaction data by **integrating** with a transaction **enrichment provider** to obtain detailed categories.*



*The core idea is to take **generic financial transactions**, each containing data such as a **transaction ID** and **account ID** and enrich them.*

*Enriched transactions must include categories such as **"Food," "Entertainment," "Groceries," "Health," "Travel,"** as well as the **merchant's name** and an optional **merchant logo**.*

*Enriched transactions will significantly **enhance** the usability of the transaction data for **end-users** by adding recognizable and transaction details.*

Scenario 1

```scala
import cats.effect.IO
import io.chrisdavenport.fuuid.FUUID
import org.http4s.{Method, Request, Uri}
import org.http4s.client.Client

// --- ADTs (Product Types) ---
case class Transaction(id: String, accountId: FUUID)

trait EnrichmentService {
  def submitTransaction(transaction: Transaction): IO[Transaction]
}

object EnrichmentService {

  def apply(baseUri: Uri, httpClient: Client[IO]): EnrichmentService = new EnrichmentService {

    def submitTransaction(transaction: Transaction): IO[Transaction] = {

      val request = Request[IO](Method.POST, baseUri / "enrich")
        .withEntity(transaction)

      httpClient.expect[Transaction](request)
    }
  }
}

```

Scenario 2

```scala
import cats.effect.IO
import cats.implicits._
import io.chrisdavenport.fuuid.FUUID
import org.http4s.{Method, Request, Uri}
import org.http4s.client.Client

// --- ADTs (Product Types) ---

case class Transaction(id: String, accountId: FUUID)

case class TripleEnrichedTransaction(
  id: String,
  accountId: FUUID,
  category: String,
  merchantName: Option[String],
  merchantLogo: Option[String],
  customCategory: Option[String]
)

// --- ADTs (Sum Types) ---

sealed trait EnrichmentStatus
case object Success extends EnrichmentStatus
case object Failure extends EnrichmentStatus

```

```scala
// --- IO-based Implementation for Triple Enrichment ---

trait TripleEnrichmentService {
  def enrich(transactions: List[Transaction]): IO[(List[TripleEnrichedTransaction], EnrichmentStatus)]
}

object TripleEnrichmentService {

  def apply(baseUri: Uri, httpClient: Client[IO]): TripleEnrichmentService = new TripleEnrichmentService {

    def enrich(transactions: List[Transaction]): IO[(List[TripleEnrichedTransaction], EnrichmentStatus)] = {

      val request = Request[IO](Method.POST, baseUri / "enrich" / "transactions")
        .withEntity(transactions)

      httpClient.expect[List[TripleEnrichedTransaction]](request)
        .map(enrichedTransactions => {
          val enrichedWithCustomCategory = enrichedTransactions.map { t =>
            t.copy(customCategory = Some("Custom Category Example"))
          }
          (enrichedWithCustomCategory, Success)
        })
        .handleError(_ => (List.empty[TripleEnrichedTransaction], Failure))
    }
  }
}
```

Scenario 3

```scala
import cats.effect.IO
import cats.implicits._
import io.chrisdavenport.fuuid.FUUID

case class Transaction(id: String, accountId: FUUID)

case class TripleEnrichedTransaction(
  id: String,
  accountId: FUUID,
  category: String,
  merchantName: Option[String],
  merchantLogo: Option[String],
  customCategory: Option[String]
)

case class BelvoEnrichedTransaction(
  id: String,
  accountId: FUUID,
  merchantReputationStars: Option[Int]
)

case class FullyEnrichedTransaction(
  id: String,
  accountId: FUUID,
  category: String,
  merchantName: Option[String],
  merchantLogo: Option[String],
  customCategory: Option[String],
  merchantReputationStars: Option[Int]
)

```

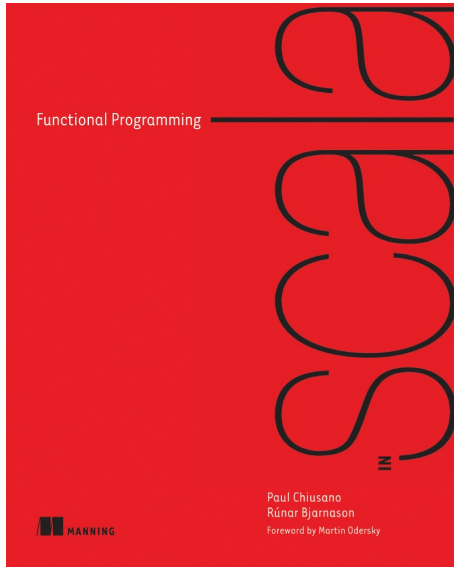
```scala
trait EnrichmentService[T] {
  def enrich(transactions: List[Transaction]): IO[List[T]]
}

class ComposedEnrichmentService(
  tripleService: EnrichmentService[TripleEnrichedTransaction],
  belvoService: EnrichmentService[BelvoEnrichedTransaction]
) {

  def enrich(transactions: List[Transaction]): IO[List[FullyEnrichedTransaction]] = {
    val tripleIO = tripleService.enrich(transactions)
    val belvoIO = belvoService.enrich(transactions)

    (tripleIO, belvoIO).parMapN { (tripleTxs, belvoTxs) =>
      for {
        tripleTx <- tripleTxs
        belvoTx <- belvoTxs.find(_.id == tripleTx.id)
      } yield FullyEnrichedTransaction(
        tripleTx.id,
        tripleTx.accountId,
        tripleTx.category,
        tripleTx.merchantName,
        tripleTx.merchantLogo,
        tripleTx.customCategory,
        belvoTx.merchantReputationStars
      )
    }.handleError(_ => List.empty[FullyEnrichedTransaction])
  }
}

```



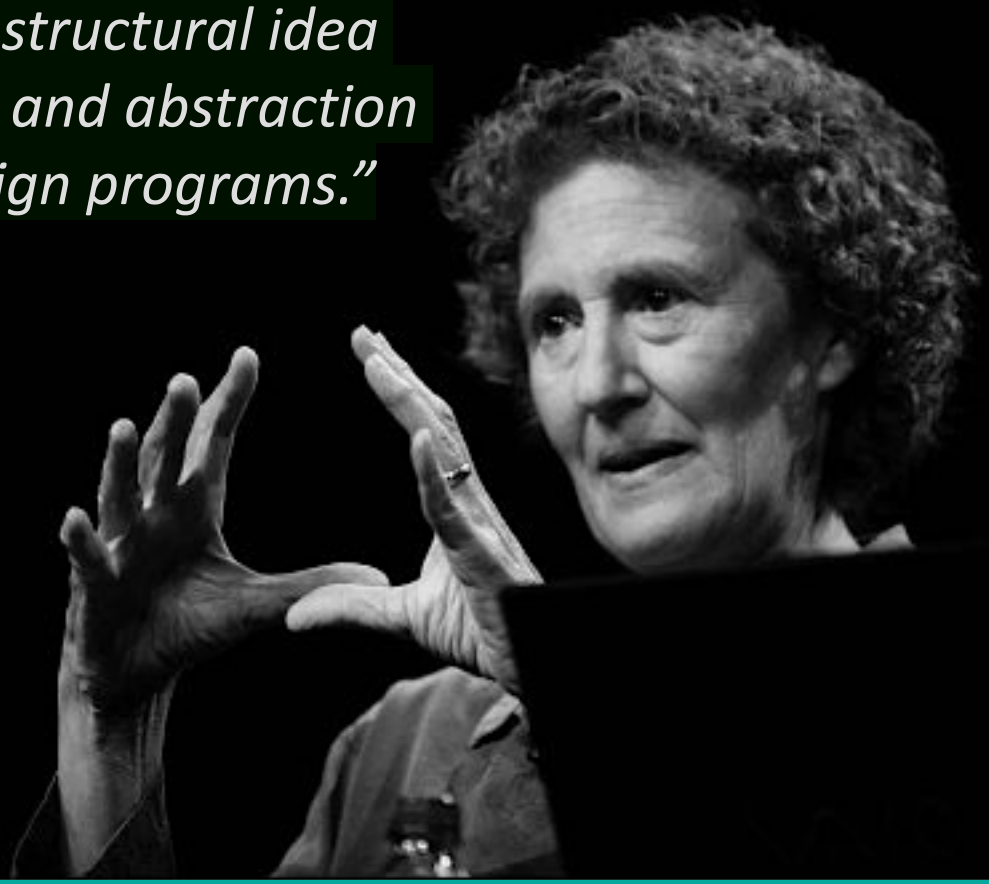
*"Functional Programming is a restriction on **how** we write programs, but not on **what** programs we can express."*
–P. Chiusano & R. Bjarnason

Recap

- What makes Scala a good addition to your skill set and tech stack?

Java 8	Java 9	Java 17
• Lambda Expressions	• Modules	• Sealed Classes

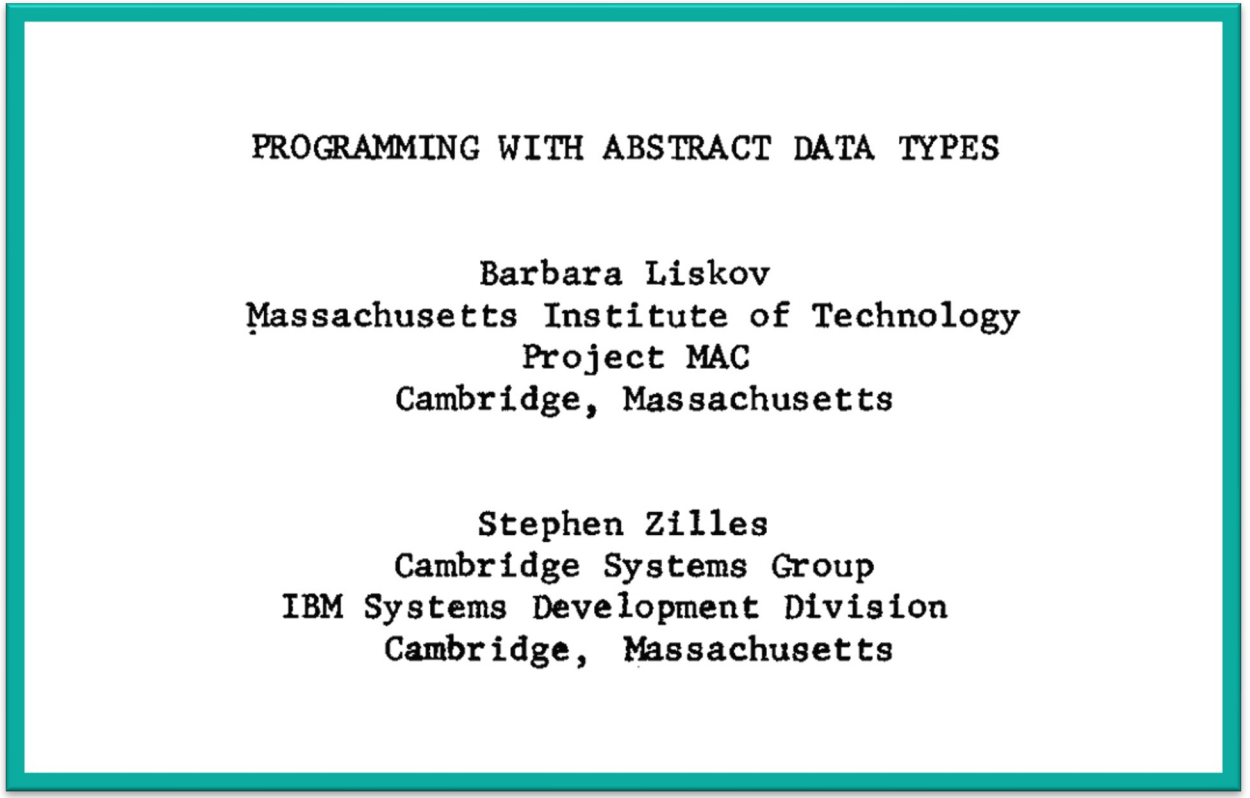
“An ADT relates a structural idea to an abstraction, and abstraction is the way we design programs.”
—Barbara Liskov



How Data Abstraction changed Computing forever
Barbara Liskov, 2019 | TEDxMIT

https://www.youtube.com/watch?v=_jTc1BTFdIo

*"This paper presents an approach which allows the set of **built-in abstractions** to be **augmented** when the need for a new **data abstraction** is discovered."*
–B. Liskov, 1974



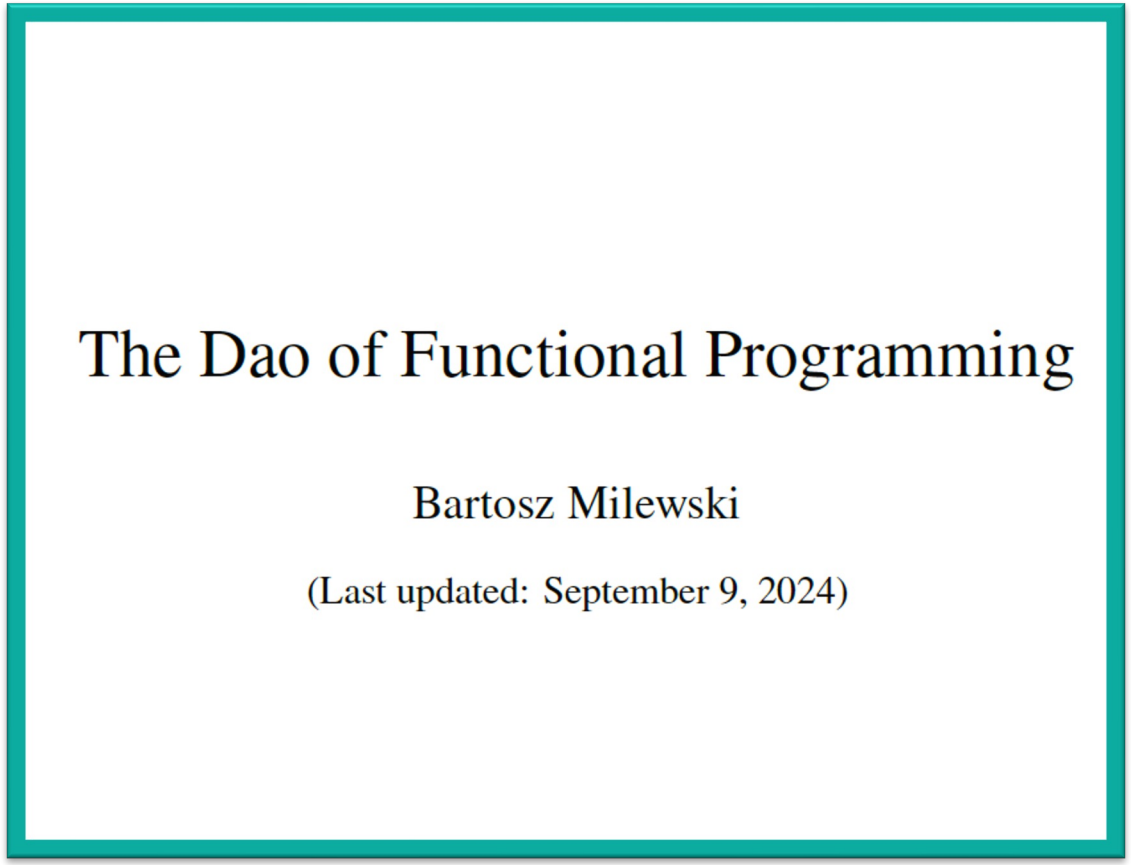
PROGRAMMING WITH ABSTRACT DATA TYPES

Barbara Liskov
Massachusetts Institute of Technology
Project MAC
Cambridge, Massachusetts

Stephen Zilles
Cambridge Systems Group
IBM Systems Development Division
Cambridge, Massachusetts

https://dl.acm.org/doi/10.1145/800233.807045

*"Programming starts with **types** and **functions**."*
–Bartosz Milewski


The Dao of Functional Programming

Bartosz Milewski

(Last updated: September 9, 2024)

https://github.com/BartoszMilewski/Publications/blob/master/TheDaoOfFP/DaoFP.pdf

Scala FP Ecosystem

Language Design

- Scala-lang
- Scalacenter

Community

Lambda World
CÁDIZ, SPAIN
02 Oct 2024 - 04 Oct 2024

Art of Scala
WARSAW, POLAND
10 Oct 2024

ScalaIO
PARIS, FRANCE
07 Nov 2024 - 08 Nov 2024

Functional Scala
LONDON, UK
30 Nov 2024 - 01 Dec 2024

Scalar
WARSAW, POLAND
27 Mar 2025 - 28 Mar 2025

- OSS Contributors and Maintainers
- Scala Ambassadors
- Rock the JVM!
- Industry Adopters

OSS Technologies and Tooling

- Typelevel
  - Cats
  - Cats Effects
  - fs2
  - htt4s
  - skunk
- ZIO
- Scalameta
  - Metals

https://www.scala-lang.org/community/

https://typelevel.org/cats/ADOPTERS.html

Thanks!