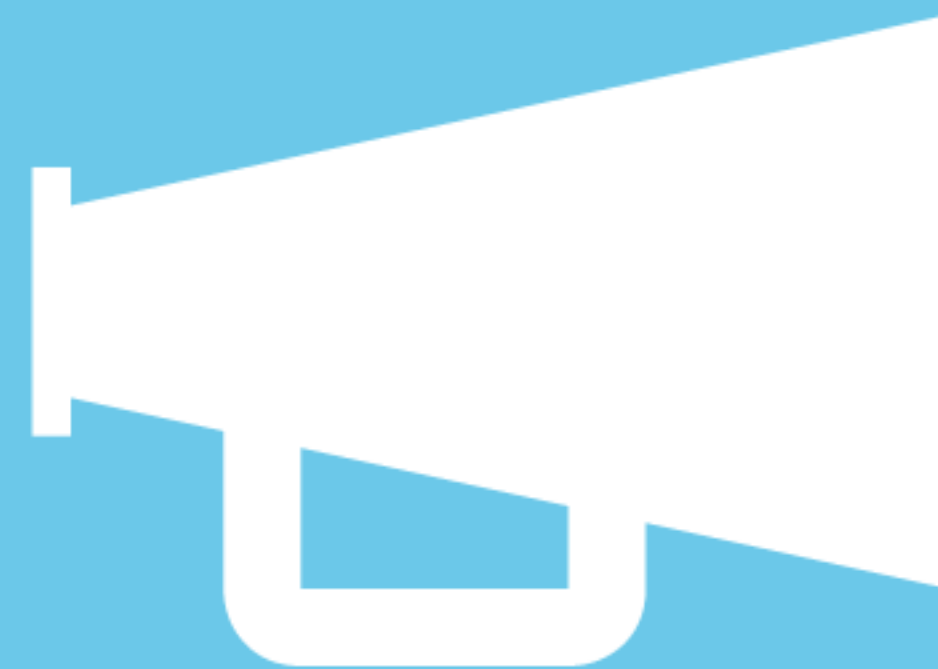
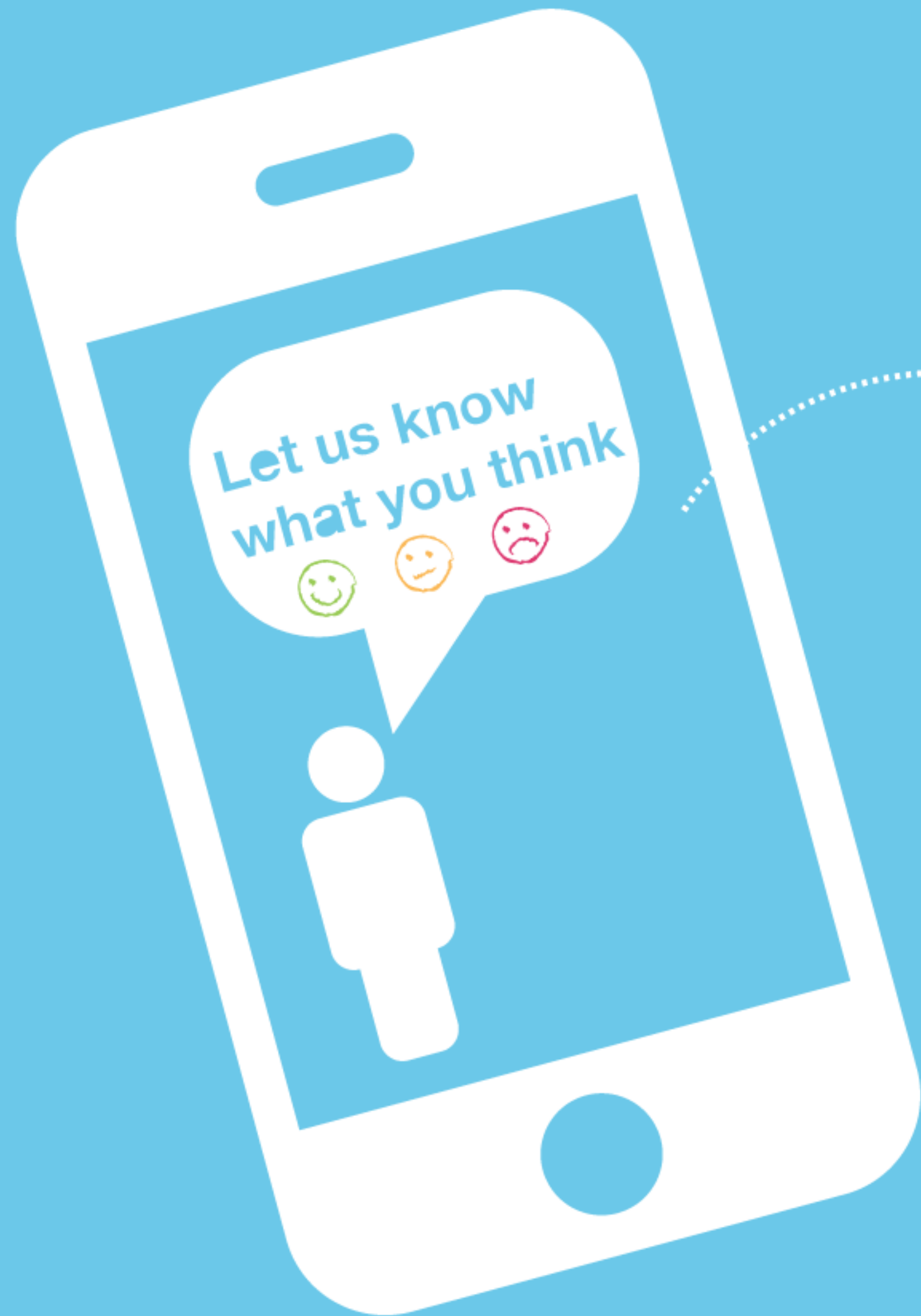




ADAM WARSKI

# TRANSACTIONAL EVENT SOURCING USING SLICK



Please use the  
Scala Days app  
to rate sessions.

## EVENT SOURCING?

- ▶ All changes in the system are captured as a sequence of events

## WHY EVENT SOURCING?

- ▶ IT: **Information** Technology
  - ▶ do not loose information!
- ▶ Audit log
- ▶ Re-create system state

# ME + AUDIT = HIBERNATE ENVERS




```
@Entity
public class Person {
    @Id
    @GeneratedValue
    private int id;

    @Audited
    private String name;

    @Audited
    private String surname;

    @Audited
    @ManyToOne
    private Address address;
```

## ME

- ▶ coder @  SOFTWAREMILL
  - ▶ Lightbend consulting partner
- ▶ mainly Scala
- ▶ open-source: MacWire, ElasticMQ, Bootzooka, ...
- ▶ <http://www.warski.org> / @adamwarski

## THE GOAL

- ▶ Get the benefits of Event Sourcing ...
- ▶ ... still being able to leverage RDBMS features
  - ▶ transactions
  - ▶ schema
  - ▶ SQL

## OTHER APPROACHES

- ▶ Event Store (<https://geteventstore.com>)
- ▶ Akka Persistence (<http://doc.akka.io/docs/akka/current>)
- ▶ Eventuate (<http://eventuate.io>)
- ▶ + more



## EVENTS

- ▶ Immutable
- ▶ Primary source of truth
- ▶ Past tense

## EVENTS

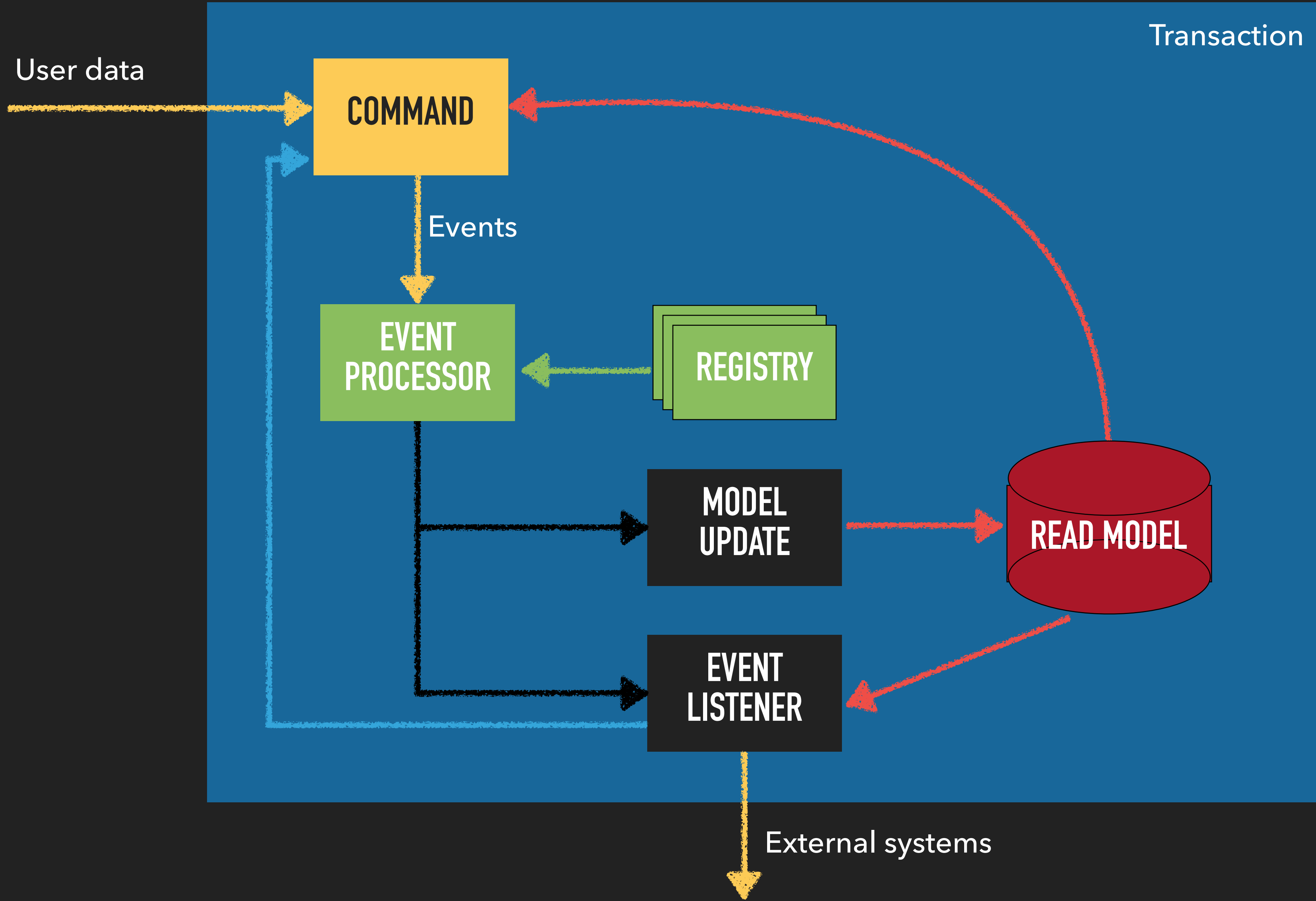
- ▶ Payload: arbitrary json (`case class`)
- ▶ Event type (name of class)
- ▶ Aggregate type & id
- ▶ Id, timestamp
- ▶ Transaction id
- ▶ User id

## HOW?

- ▶ Events are stored in a dedicated table
- ▶ Basing on events, a read model is updated
  - ▶ similar to what a "traditional" CRUD model could be
- ▶ Consistency: both done in a single transaction

# SLICK

- ▶ We operate on `DBIOAction[T]`
- ▶ a **description of actions** to be done
  - ▶ execution deferred later
- ▶ can be sequenced using `flatMap`
- ▶ yes, it's a Monad



**EVENT SOURCING LIVE**

## POTENTIAL PROBLEMS

- ▶ Ordering of concurrent events operating on the same aggregate root
- ▶ `DBIOAction` "leaks"
- ▶ Future wrapping

## SUMMING UP: FUNCTIONS INVOLVED

- ▶ `Commands: Data => CommandResult[S, F]`  
that is, `DBIOAction[(Either[S, F], List[Event])]`
- ▶ `Event listeners: Event => DBIOAction[List[Event]]`
- ▶ `Model updates: Event => DBIOAction[Unit]`



## LINKS

- ▶ <https://github.com/softwaremill/slick-eventsourcing>
  - ▶ README+blog longer than code
  - ▶ only a skeleton
  - ▶ probably for customisation
- ▶ <https://github.com/adamw/slick-eventsourcing-pres>

**THANK YOU!**



*Please*

**Remember to  
rate this session**

*Thank you!*

