CQRS

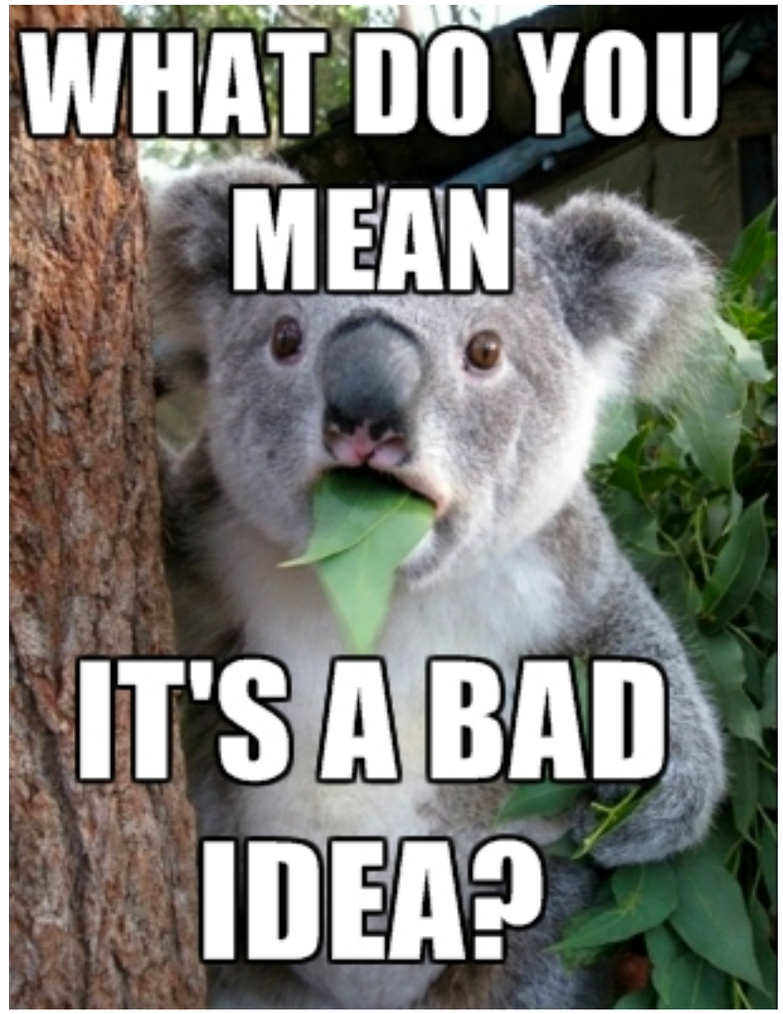
explained with a million dollar idea

What are we going to do?

- The million dollar idea
- Command and query segregation
- Command and query responsibility segregation
- Code structure
- Disclaimer: when (not) to use

The million dollar idea

- Based on my graduation project
  - Car/ride-sharing application
  - Focus on social aspect
    (you both know this person
    or both sharing the same interest)
  - Primarily used for scheduled travel



- Assumptions
  - We need CQRS

Command and query segregation

- Commands
  - Changing the state of a system without returning a value
- Queries
  - Return a result without changing the state of the system (generating no side-effects)

- Can be described as use-cases, i.e.
  - RegisterRide
  - FindRide
  - FindRideMatch

Why?

- Ease of mind, you don’t have to worry about executing queries since you “can’t” break things

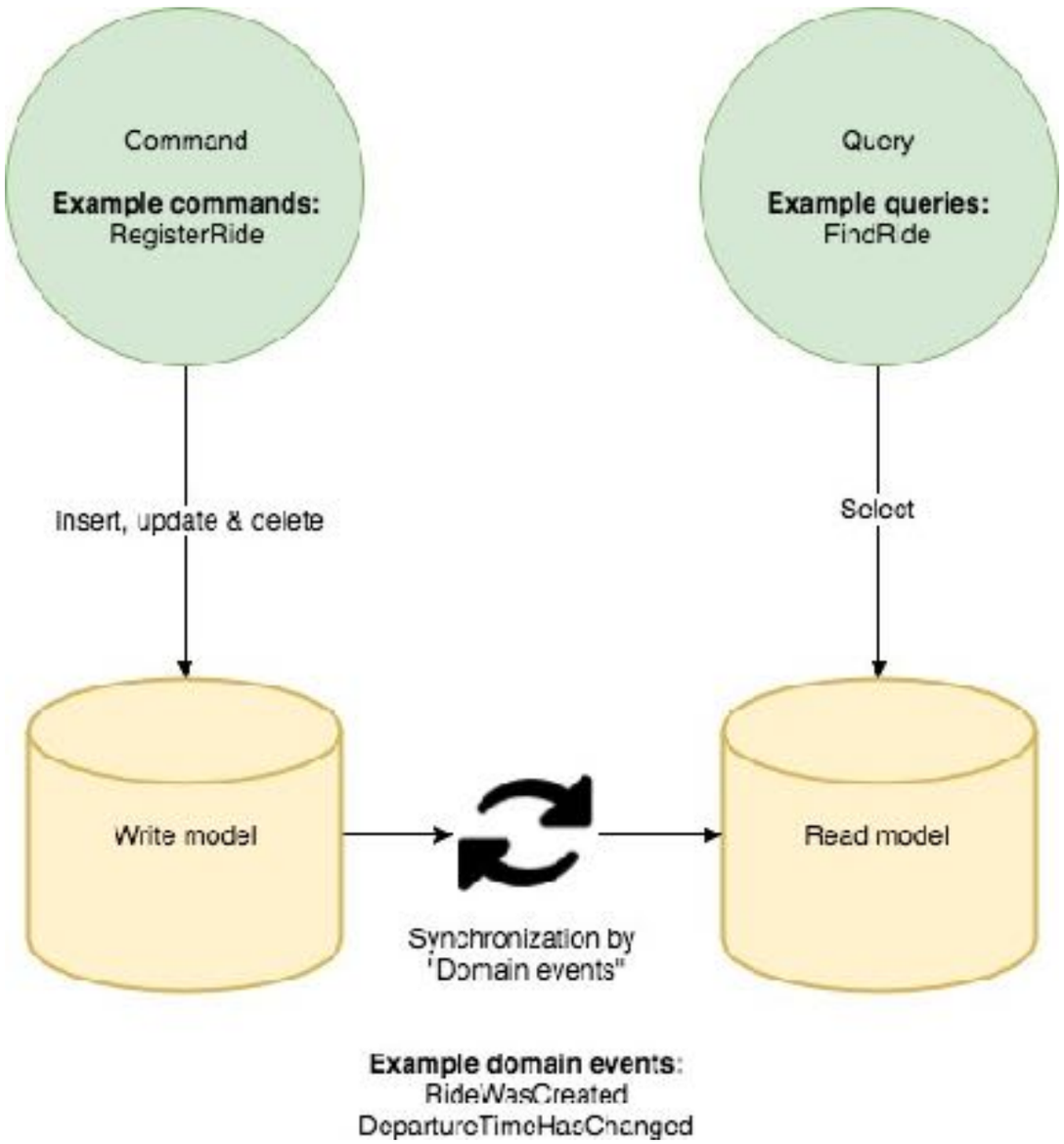
- “Out of the box” able to perform commands asynchronously

Command and query responsibility segregation

"At its heart is the notion that you can use a different model to update information than the model you use to read information."

–Martin Fowler

Command and query responsibility segregation



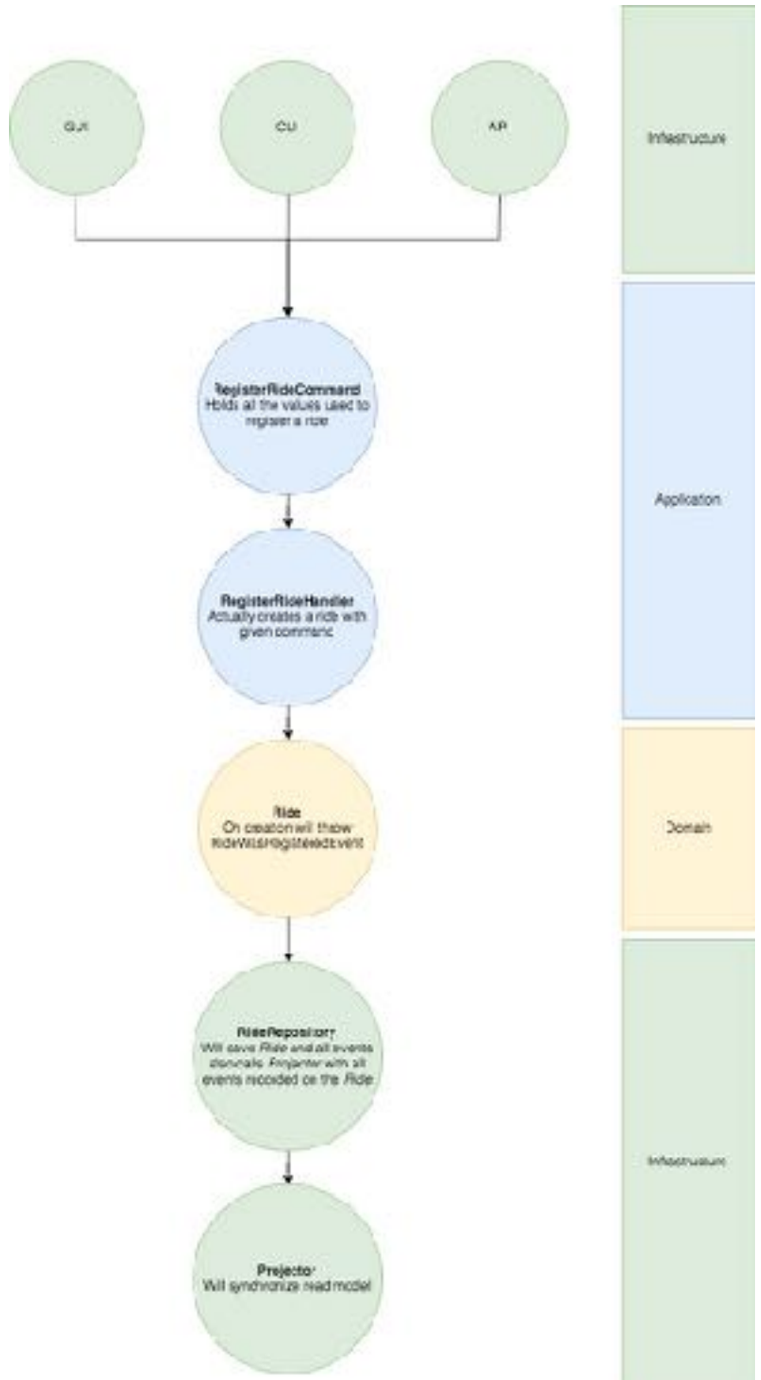
Command and query responsibility segregation

- Different read and write models
- Domain events
  - They describe what happend in the domain (for example RideWasCreated, DepartureTimeHasChanged)
  - Are dispatched for every change in the domain
  - Are used to update the read models
  - Side effect: allows traceability and accountability (i.e. the bug is caused by this sequence of steps)

Why?

- Storing data at the best possible place (No-SQL, Graph etc) in the best possible format (normalized or denormalized) for the use-case
- Scalability advantages
- Better performance

The example application



Disclaimer: when (not) to use?

- Be aware that CQRS can overcomplicate your application, explore alternatives before falling for the “cool” factor of this pattern
- Consider using CQRS in the following cases
  - When working in large teams
  - When working with difficult business logic
  - When scalability matters

Further reading…

- https://leanpub.com/ddd-in-php - DDD in PHP

- http://getprooph.org/ - The CQRS and event sourcing components for PHP