On Deck App

Musings on a passion project

About NRASC

- Community volunteer organisation since 1932
  - Inclusive, multigenerational
- 30-50 members, age 3-80+
- 7 am - 9:30 am Sundays 6 mo.
- Meet = Events and Heats
- Event = eg. 50m breaststroke hcp
- Race types (handicap, scratch, relay)

How to run a swimming race

* Officials:
  * Starter (to gather swimmers and start the race)
  * Timekeepers (one per lane with a stopwatch)
  * Invigilator (to judge placings)
  * Time recorder (to collect all of the results)

"Legacy"

- Excel-based, paper on clipboards, CC
- Handicapped starts calculated by hand
- Registrations by hand, break into heats
- One copy at Starting Block, one at Finish
- Struggle to finish meet on time
- Post-meet manual data entry

… before you begin - regulatory

* Get the OK from your employer to work on your side project, even if you are working on your project in your own time
    * Don't use company equipment
* Get the OK from your significant other to spend lots of time
    * Spend ~~leisure~~ cricket-watching time coding
    * I don't recommend it, but a lockdown or 3 can give you spare time

Preparation

* Gather requirements - many processes are done by hand, with unspoken rules
* Build trust and get buy-in
  * Convince people who have done things this way for many years
  * Respect their experience
  * Find out why things are done this way before changing it
  * Show that you understand the club and its culture and rituals
* Consult extensively
* Iterative development cycles gathering feedback

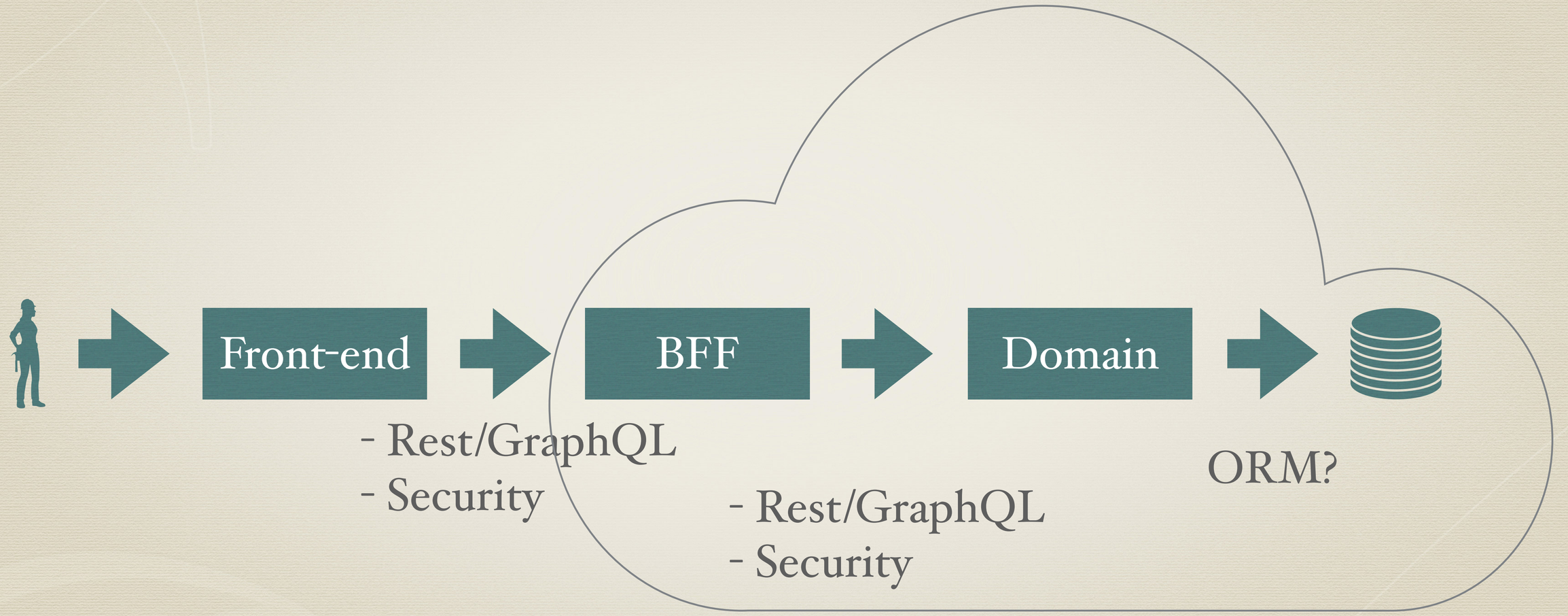
On Deck App - requirements

* I had to have a punny name
* Restrict scope (just runs swim meets, not membership or payment)
* iPad/tablet operation (web single-page app)
* automate as much as practical but let Administrator stay in control
  * GUI developed with all officials, iteratively
* RBAC
* Documentation - did you think you'd get away without it?

Lessons learnt

* Extremely rewarding project, helped me rediscover the joy of code
* Don't have time to do it properly? Do it twice
* Development priorities for a single-dev project:
  * Don't scale up for traffic, try to scale up dev time vs outcome
    Minimise friction:
    * avoid REST/GraphQL
    * avoid self hosting and explicit hosting

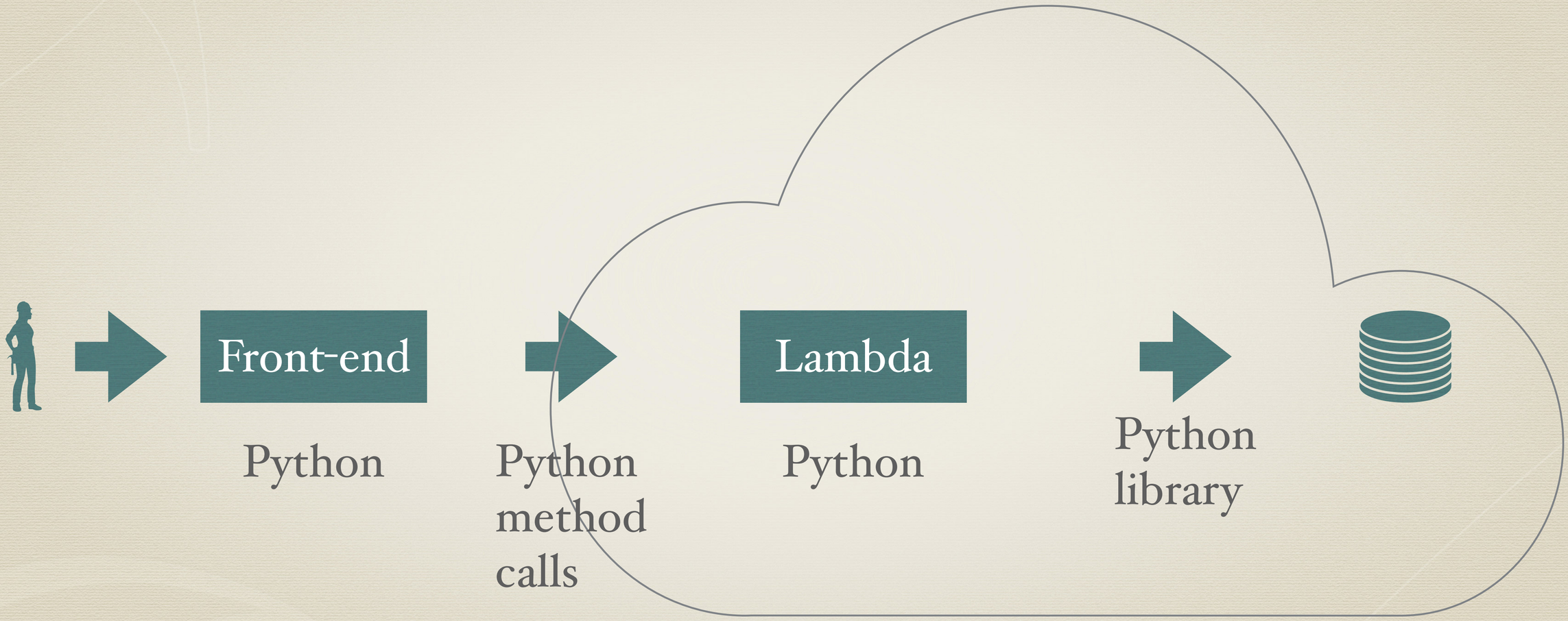
Classical Enterprise Architecture



Platforms investigated

* Elm / Lamdera by REA alumni Mario Rogic - is Elm a dead-end?
* Version 1 of my app was in https://anvil.works
  * Full-stack Python, extremely low barrier to entry
  * Hosted and developed on Anvil's site
  * Slow database access killed this version
  * Python codebases need an extensive test suite - no compiler to save you

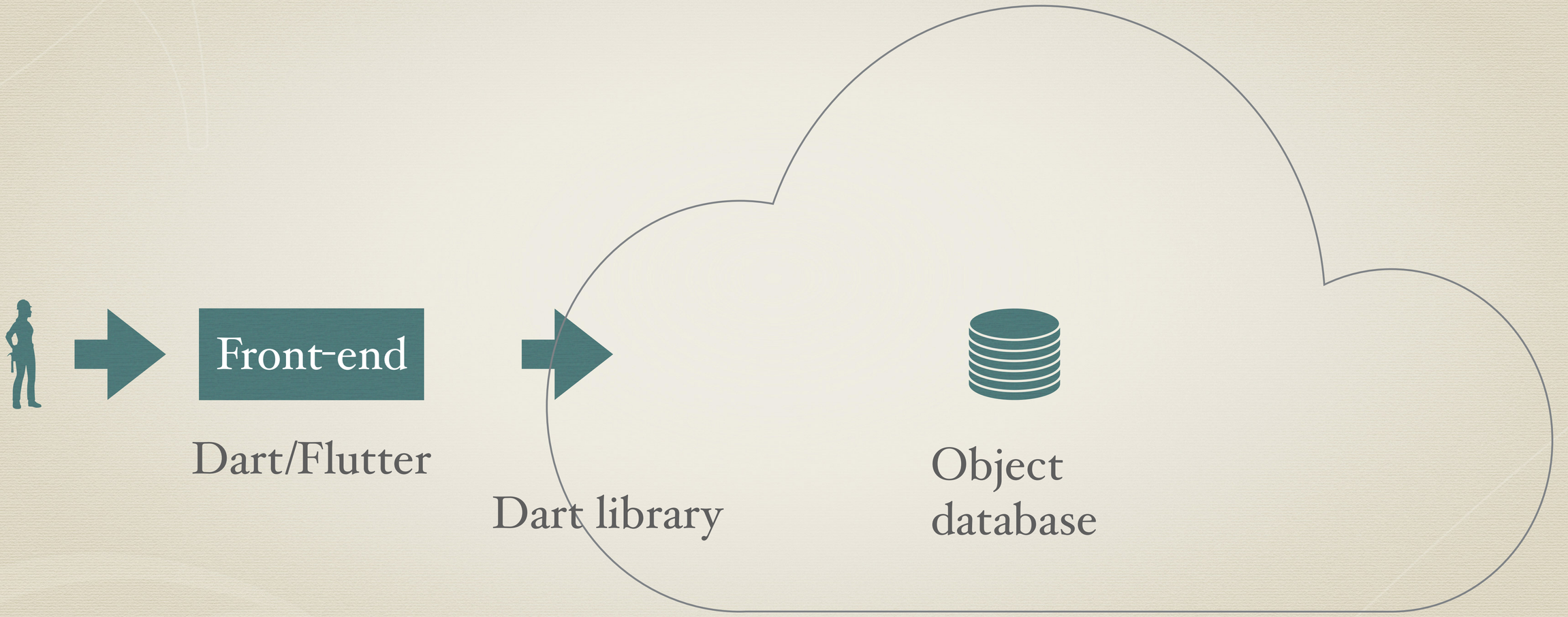
anvil.works


Front-end
Python
Python method calls
Lambda
Python
Python library

Current version

* Dart/Flutter with Firebase for db and auth
* CI/CD using github actions
* Option to develop natively
* Relies on Google to keep the lights on

Dart/Flutter/Firebase/Firestore



What I've enjoyed

* *Problem solving*, eg
  * 10 swimmers, 4 available lanes
  * Invent algorithm to choose relay teams to minimise starts
  * Create a GUI to match users' mental models
    * Extensive walk-throughs with prototypes
    * Find many bugs, many friction points
* Seeing the impact of my work firsthand
  * Positive feedback

What has been... less fun

* Pressure when building for 50 friends is surprisingly high, immovable deadlines
* Code maintenance and bug-tracking, prioritising and fixing
* Bug fixing at 7am in a swimming costume with 50 people waiting
* Dealing with spotty internet at the pool

Finally

* Go out and do good