



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Angular Testing

## 2 - E2E Advanced

# Agenda

1. Misc. Features
2. Patterns for E2E Tests & Databases



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Agenda

1. **Misc. Features**
2. Patterns for E2E Tests & Databases



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Mocking Requests

```
cy.intercept('GET', 'http://localhost:4200/holidays', {  
  body: {  
    holidays: [  
      {  
        title: 'Cambodia',  
        teaser: 'Discover old temples and learn about the great Khmer Empire',  
        imageUrl: 'https://eternal-app.s3.eu-central-1.amazonaws.com/assets/AngkorWatSmall.jpg',  
        description:  
          'Travel to Siem Reap in Cambodia and visit the...'  
      }  
    ]  
  }  
});
```



# Asserting Requests

```
it('should assert the holidays request', () => {  
  
  cy.intercept('https://api.eternal-holidays.net/holidasy').as('request');  
  
  cy.visit('');  
  
  cy.testid('btn-holidays').click();  
  
  cy.wait('@request');  
  
});
```



# Creating Requests

`cy`

```
.request('https://api.eternal-holidays.net/holiday')  
  
.then((res) => cy.log(res.body));
```

- All HTTP Methods available
- Can send to any origin
  - No Same-Origin Policy
- No CORS
- Good fit for "Test-APIs"

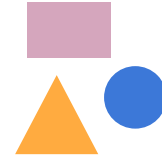


# 3 Level Architecture

Tests



Page Object Models



Utility Functions



# Page Object Model

```
class Sidemenu {  
  click(name: "Customers" | "Holidays"): Chainable {  
    cy.get("mat-drawer a").contains(name).click();  
  }  
}  
  
export const sidemenu = new Sidemenu();
```





# Cypress.Commands.add

- Extends the cy object
- Needs to be done two times (TS declaration and implementation)
- Combines existing cy commands
- No need to return data



# Cypress Commands Example

```
declare namespace Cypress {  
  interface Chainable<Subject> {  
    testid(selector: string): Chainable<jQuery<HTMLElement>>;  
  }  
}
```

```
Cypress.Commands.add(  
  'testid',  
  (selector: string) => cy.get(`[data-testid=${selector}]`)  
);
```



# Cypress.Commands.addQuery (since v12)

- As Command but marks it as query
- Enforces Retry-bility
  - Synchronous
  - Retried
  - Indempotent
    - don't use side effects!

```
Cypress.Commands.addQuery(  
  'Testid',  
  function (selector: string, options = {}) {  
    const getFn = cy.now('get', `[data-testid=${selector}]`, options)  
    as (subject: unknown) => unknown;  
  
    return (subject) => {  
      const element = getFn(subject);  
      return element;  
    };  
  });
```



# cy.task 1/2

- Run commands in Node.js
- Far more possibilities:
  - Writing to the filesystem (logging)
  - Connecting to a database
  - Combine Cypress with other E2E Tools



# cy.task 2/2

```
export default defineConfig({
  e2e: {
    setupNodeEvents(on, config) {
      on('task', {
        async log(message: string) {
          await fs.writeFile('cypress.log', message);
        },
      });
    },
  },
});
```



# Agenda

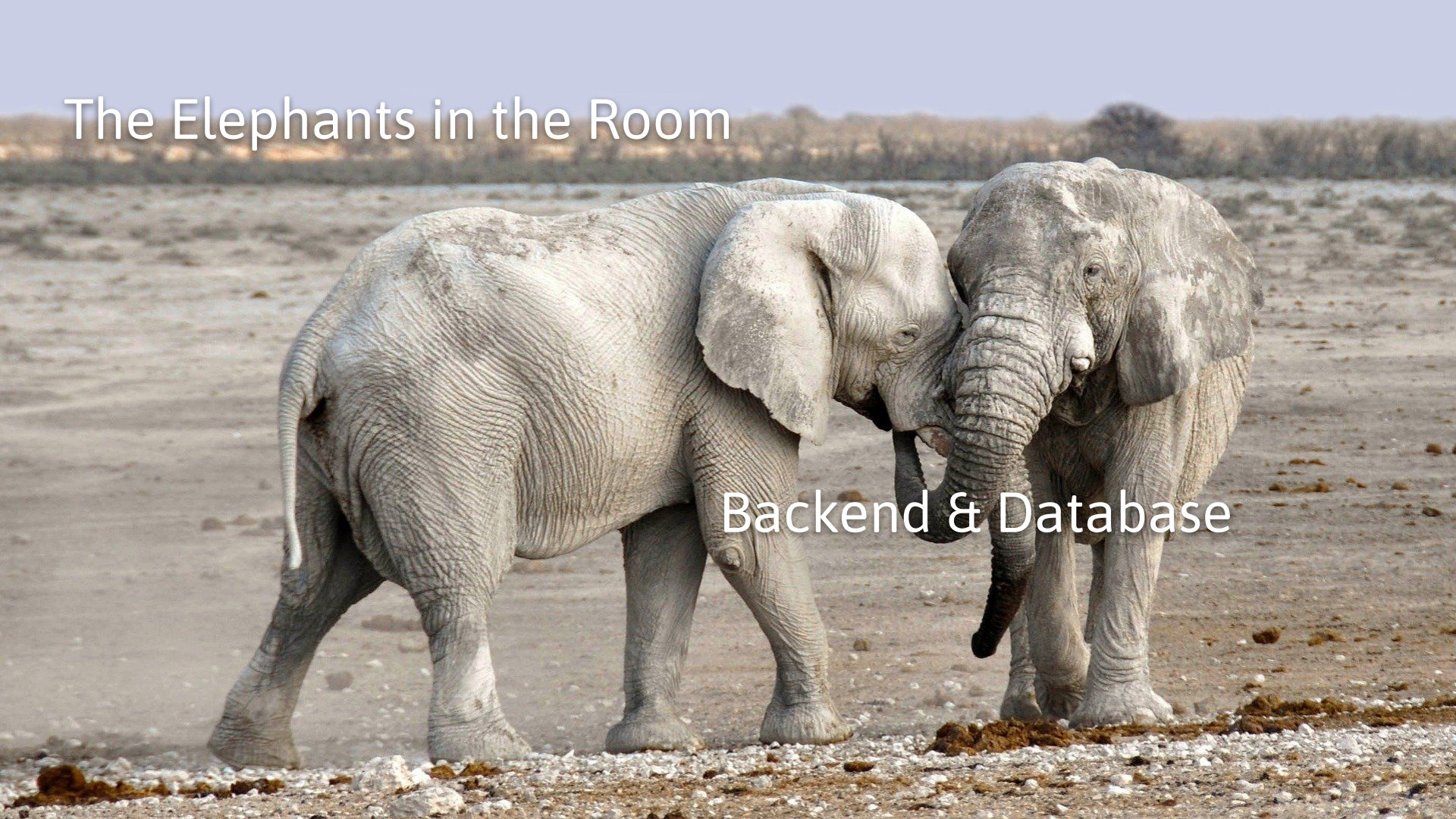
1. Misc. Features
2. **Patterns for E2E Tests & Databases**



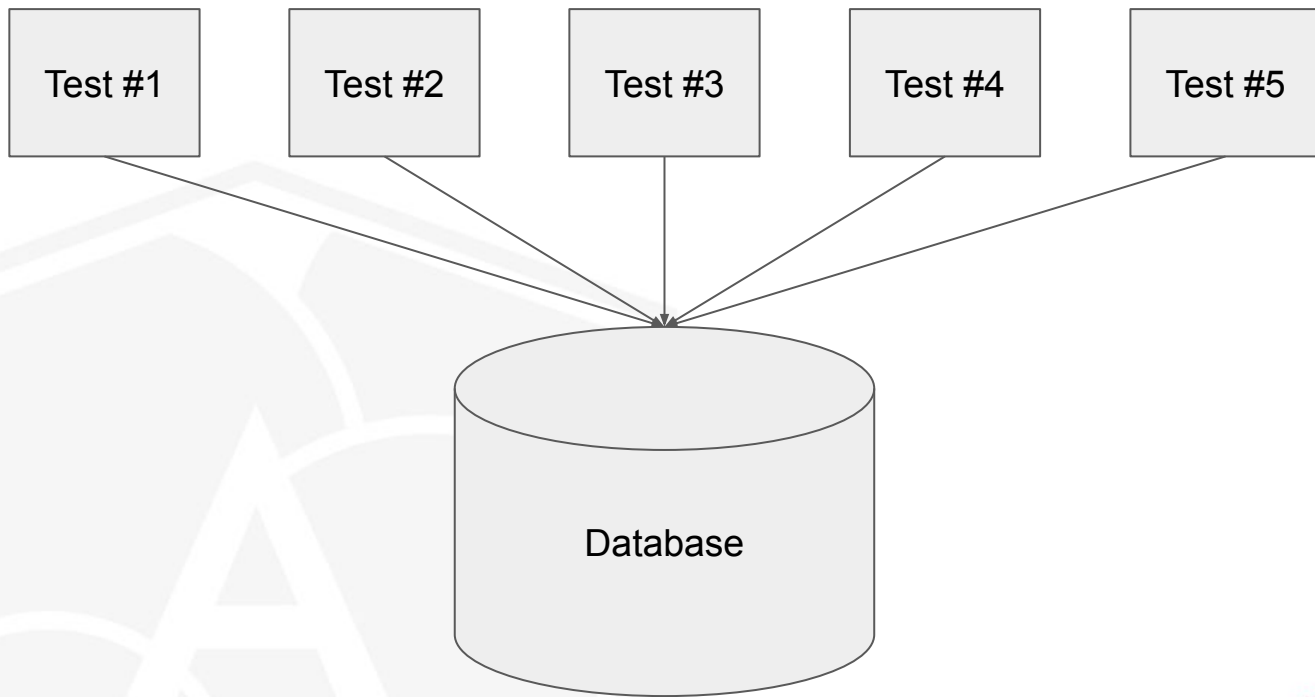
ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# The Elephants in the Room

Backend & Database

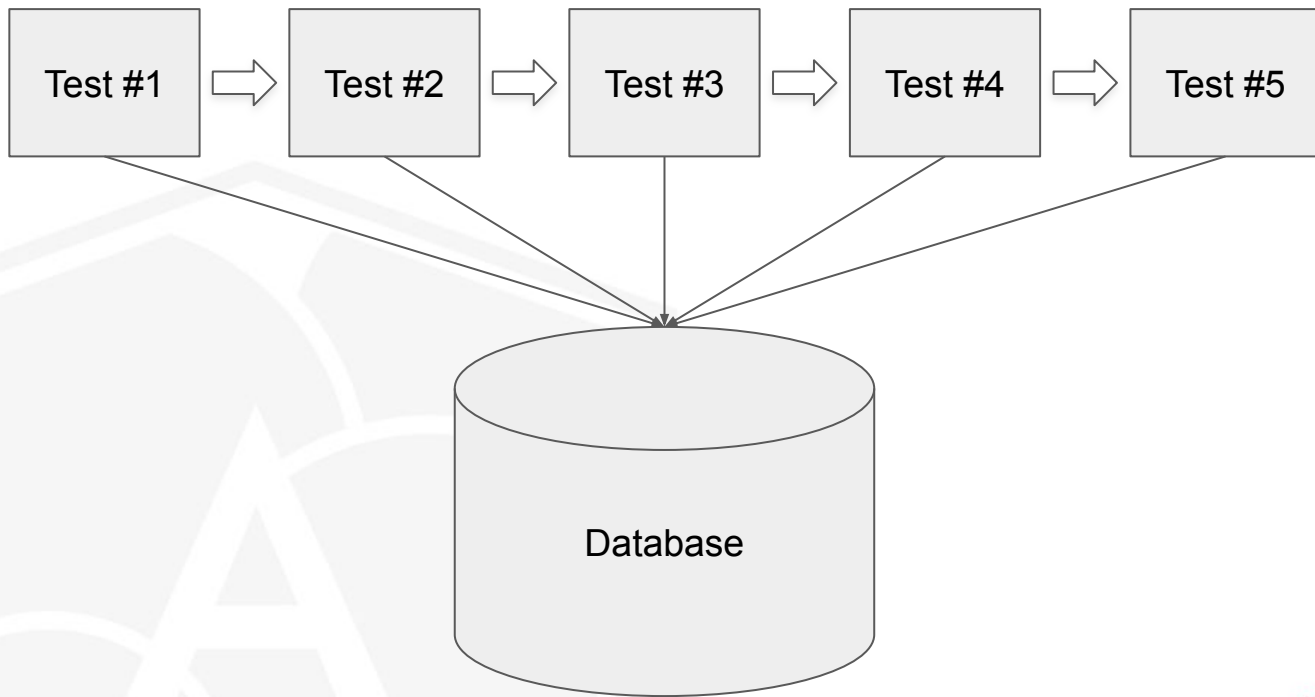




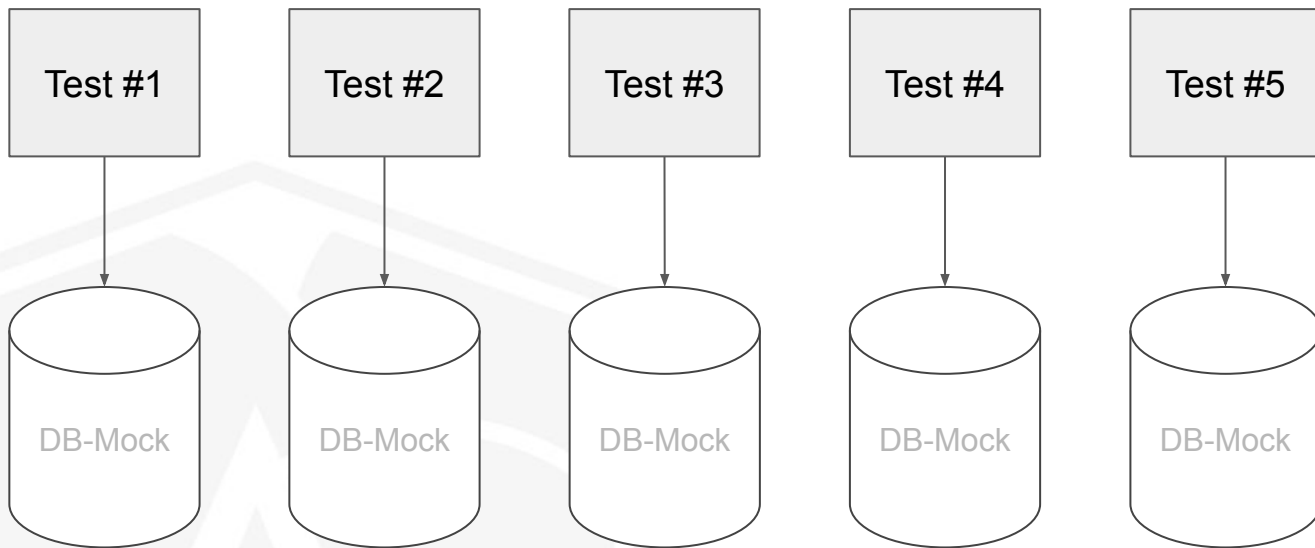




## Indirect "Transitive" Coupling



## Copy Strategy from non-E2E???



Database in E2E Tests



Database in Non-E2E Tests

Test Seeded Database & done???



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Issues with Test Seed

- "One size fits all" approach
- Tight Coupling → Not scalable
- Fast Reseeding not always possible
- Multiple Databases
- Data from External Systems → no Seeding possible

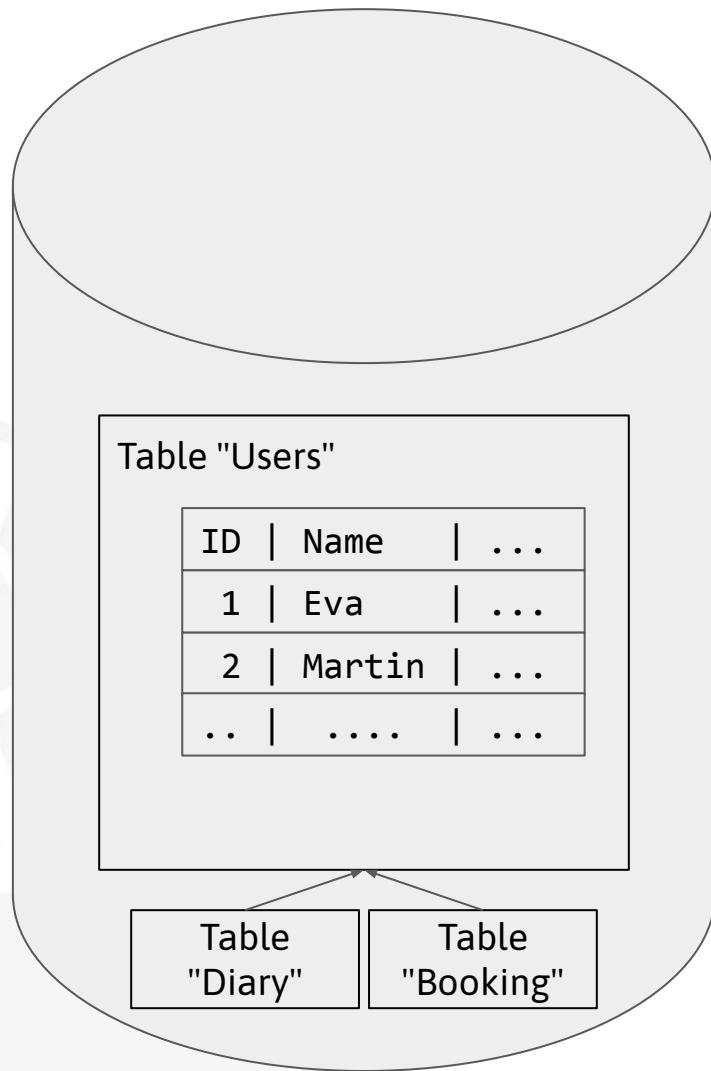


# Individual Scope - Best Case Scenario

- Data is referenced to a particular entity
  - User
  - Product
  - ...
- Multi-Tenant Systems
- Customer-Centric Systems
  - Insurances
  - Banks



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE



Test 1

"Add Diary"

Table "Users"

ID	Name	...
1	Eva	...
2	Martin	...
3	Max	...
..	....	...

Table  
"Diary"

Table  
"Booking"



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE



Test 1

it('should add a  
new diary', ...);

Table "Users"

ID	Name	...
1	Eva	...
2	Martin	...
3	Max	...
4	Lucy	...
..	....	...

Table  
"Diary"

Table  
"Booking"

Test 2

it('should start  
the tutorial on  
empty diaries',  
...)



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE

Test Setup !== Test



# API Arrange Possibilities: Normal Requests

- Default Case
- Call same endpoints as the Frontend
- Don't use the frontend directly!
- `cy.task()` as alternative



# API Arrange Possibilities: Dedicated Test API

- Backend provides special API for test mode
- Shortcuts possible, e.g.
  - merge chain of requests into one
  - Overcome Security Issues
- Best Option



# Testing Scopes

- Individual Scope
  - All data depends on a certain ID
  - e.g. Personalised Data
  - Best Option in Combination with Test API (Sign Up & In)
- Global Scope
  - Tests Affect each other
  - Challenging Parallel Runs
  - Not so easy to solve...



# Global Pattern I: Independent Tests

- Read-Only Character
- No Arranging required
- Rely on Test Seed
- Smoke Tests
- Tests for Static Elements



# Global Pattern II: Intelligent Tests

- "I'll create and find it"
- Flexible
- Requires more code



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Global Pattern III: Dependent Test Group

- Default Group
- Logical Group of Unit Tests
- Internal knowledge about other tests
- Order is important
- Database Reset after each Group Run





# Global Scope IV: Simulated Individual Context

- Mock all APIs
- Transforms a global into an individual context



# Global Scope V: Integration Tests

- Don't test it all and rely on integration tests



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

## Arrange

## Act & Assert



Test API

Application API

Test Seeded  
DB



### Individual Scope

### Global Scope

Independent

Intelligent

Dependent  
Test Groups

Simulated  
Local Context



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE

# Summary

- You will not have completely isolated tests
- Try to minimize loose coupling
- Always prefer Backend API over Test Seed
- Look out for opportunities with individual scope

