

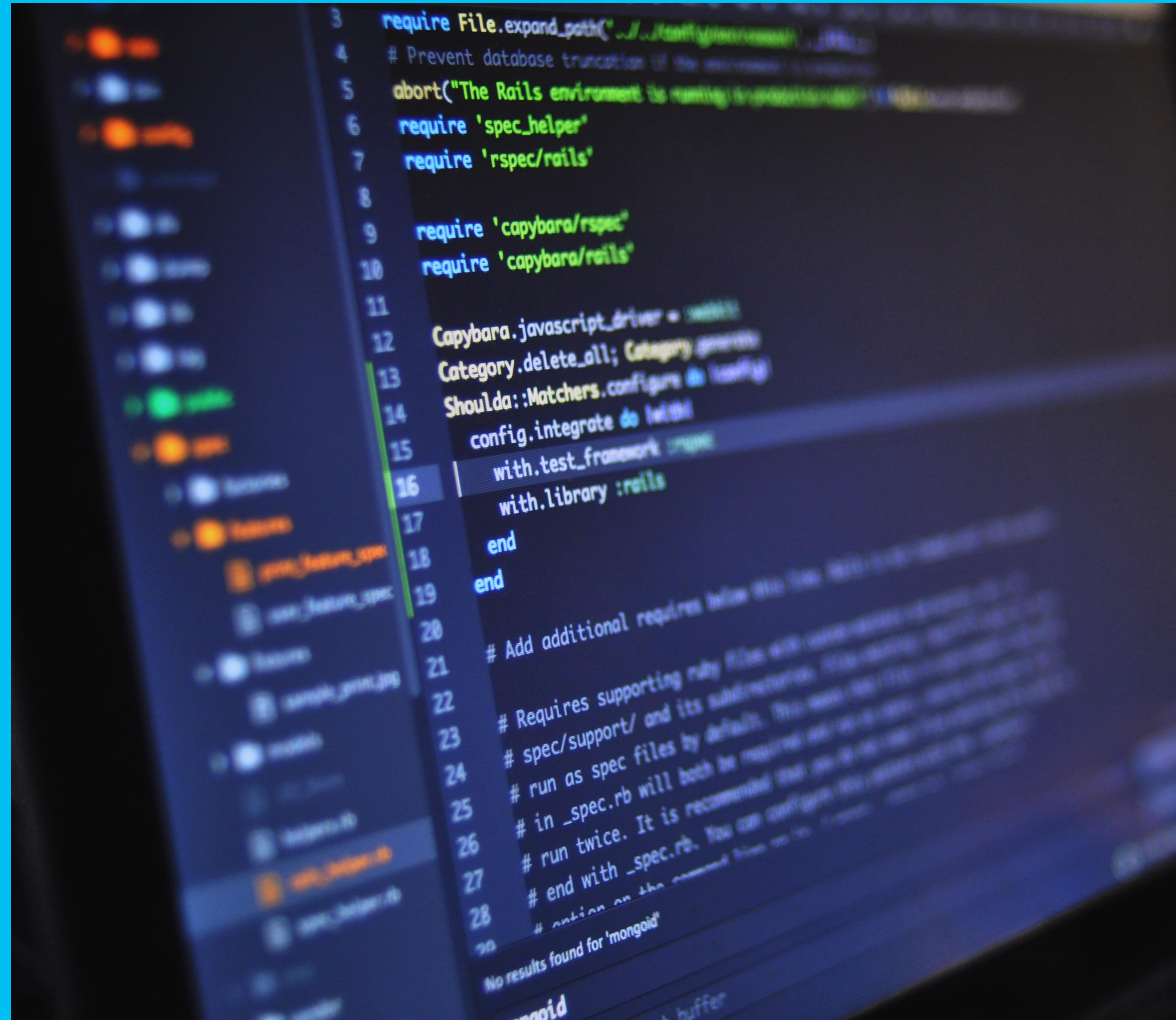
GIANFRANCO CAUTIERO - 12-05-2022

JAVA SPRING BOOT && MICROSERVICES

SPRING BOOT & MICROSERVICES

WELCOME

- Who am I?
- Who are you?
- What do you expect to learn?
- Set-up upcoming 2 days



WHO AM I?

- Gianfranco Cautiero
- Studied Computer science in Italy
- Software engineer @CodeNomads
- Java backend expert

WHO ARE YOU?

WHAT DO YOU EXPECT TO LEARN?

SET-UP COMING 2 DAYS

- Day 1: Introduction - Basics of Spring Boot
- Day 1: ~Start to build your first micro service for user api

GOALS

- Basis features of Spring
- Build simple applications using Spring
- Test driven development
- Spring is so big: where to find answers?

DAY 1 - MORNING

9:00 - 9:30 Introduction

09:30 - 10:15 Dependency Injection && Inversion of Control

10:15 - 11:00 Exercise: “Set-up a spring boot application”

11:00 - 11:15 Coffee

11:15 - 12:00 What are REST endpoints?

12:00 - 12:45 Lunch


INVERSION OF CONTROL

- With Inversion of Control we want to decouple some dependencies
- Goal is to improve simplicity
- Goal is to improve testability

DEPENDENCY INJECTION

“Dependency Injection (DI) is a form of inversion of control, where implementation are passed into an object by an IoC container”

Different types of Injection types:

- Field-based Injection
- Method-based Injection
- Constructor-based Injection ← 

DEPENDENCY INJECTION

```
class FieldBasedInjection {  
    private Object field;  
}
```

DEPENDENCY INJECTION

```
class MethodBasedInjection {  
    private Object field;  
  
    public void setField(Object toSet) {  
        this.field = toSet;  
    }  
}
```

DEPENDENCY INJECTION

```
class ConstructorBasedInjection {  
    private Object field;  
  
    public ConstructorBasedInjection(Object field) {  
        this.field = field;  
    }  
}
```

WHAT ARE ANNOTATIONS?

- “Standard way to add metadata to Java classes”
- Injection with Spring is done with `@Autowired`
- `@Autowired` tells spring the dependency needs to be instantiate, Spring will take care of the rest
- Code example

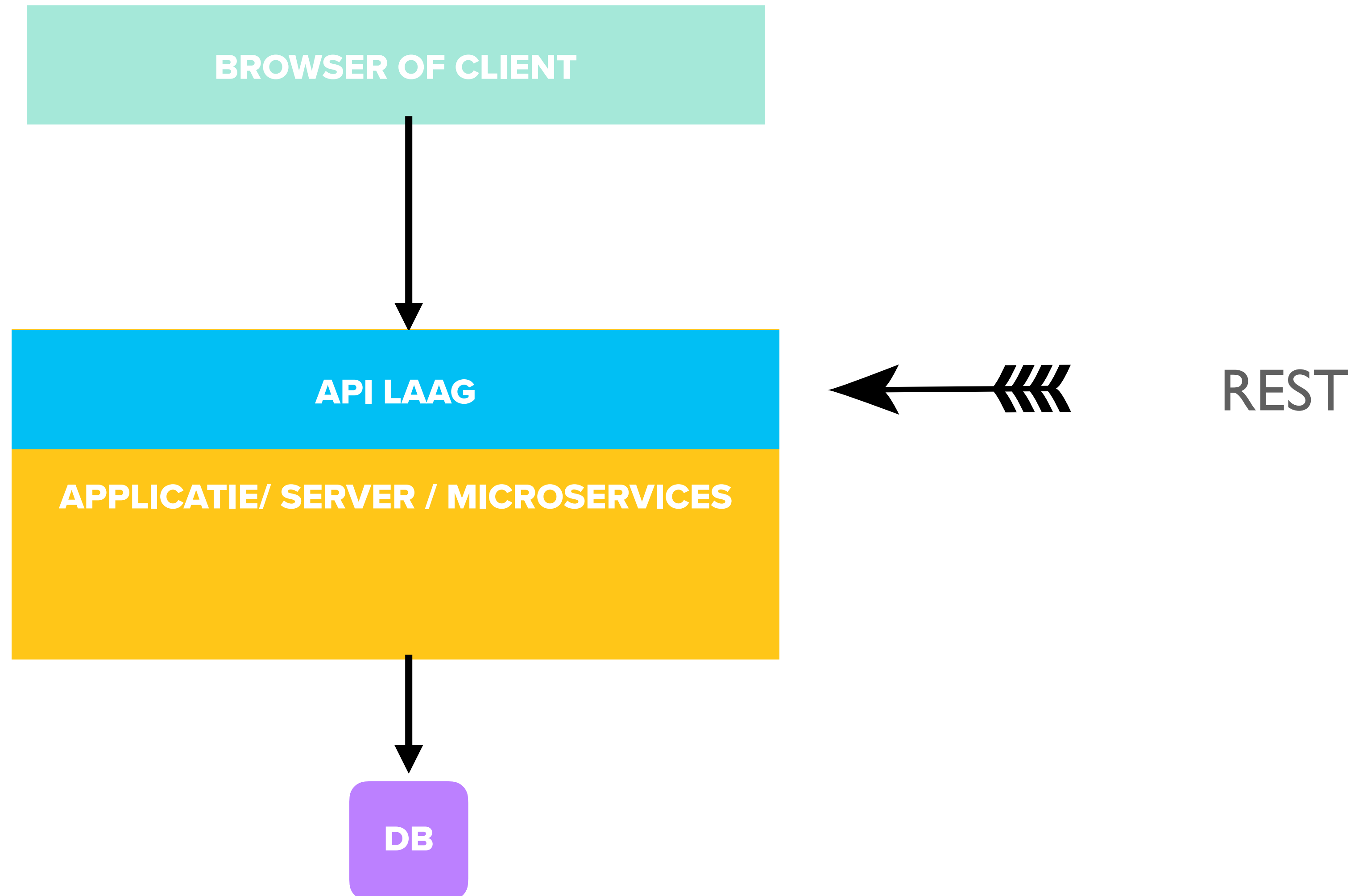
REST ENDPOINTS

- Wat is http?
 - Hypertext transfer protocol
- URL
 - Uniform resource locator

WHAT ARE REST ENDPOINTS?

- **REST: Representational State transfer**
- **Afspraken over hoe we HTTP communicatie doen**
- **Afspraken hoe een API er uit hoort te zien**
- **Wat is een API?**
 - **Application Programming Interface**

CLIENT - SERVER



REST

RESOURCE

USER

Save

Find

Delete

Modify

REST VERBS

GET	Ophalen resources
POST	Aanmaken resources
PUT	Updaten of vervangers resrouces
PATCH	Past een resource aan
DELETE	Verwijderen van een resource

REST ENDPOINT

- URL

- VERB

- “<HOST>” + “/users” GET -> Retrieve all users

DAY 1 - AFTERNOON

- 12:45 - 13:30** **Exercise: “Set-up a REST endpoint”**
- 13:30 - 14:00** **What are errors en how to handle them?**
- 14:15 - 14:30** **Exercise: “Implement ErrorHandler”**
- 14:30 - 14:45** **What are tests?**
- 14:45 - 15:05** **Coffee**
- 15:05 - 15:50** **Exercise: “Adding tests”**
- 15:15 - 16:00** **Other types of tests + Live coding + Questions**

REST BUILD UP URLS

- Always plural paths
- `/users/{name_of_user}/addresses`
- No verbs!
- Respect the REST verbs!

ERROR HANDLING - BEST PRACTICES

- **400: Bad Request**
 - **401: Unauthorised**
 - **403: Forbidden**
 - **404: Not Found**
 - **412: Precondition failed**
 - **500: Internal Server Error**
 - **503: Service Unavailable**
-
- **Bij Spring: Exceptie is altijd een 500 behalve als we het anders configureren**

ERROR HANDLING - RISKS

- Hide information (security)
 - Balance between inform the client and protecting data structures
 - Agreements in a company about them and how to react on them
-
- Difference between 404 en 503

WHAT ARE TESTS?

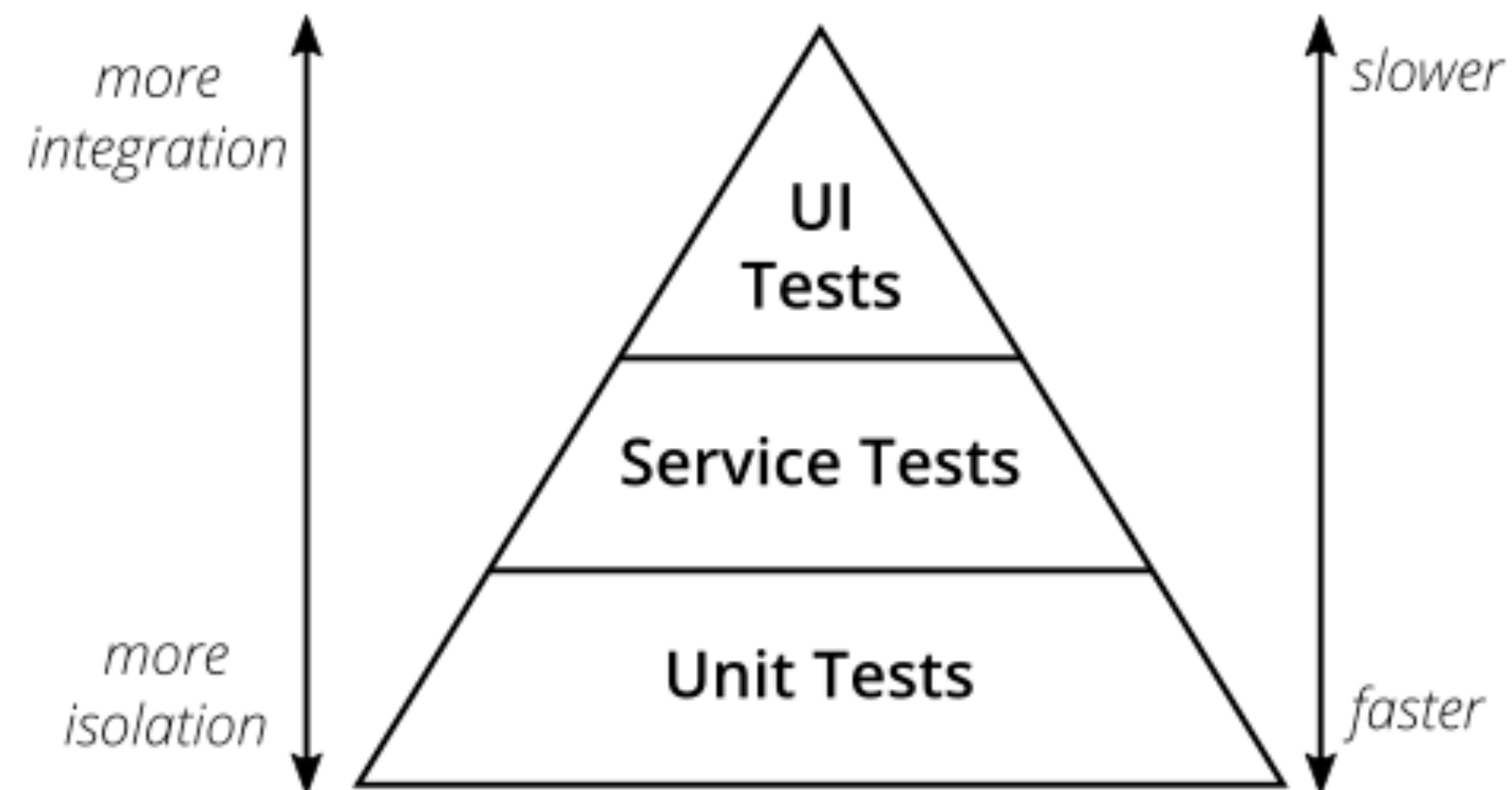


Figure 2: The Test Pyramid

<https://martinfowler.com/articles/practical-test-pyramid.html#TheTestPyramid>

HOW CAN SPRING HELP?

- **Start up the application**
 - **Initialise all the beans**
 - **Run the automated tests**
-
- **When to run the tests?**
 - **What is test coverage?**

DAY 2 - MORNING

- 9:00 - 9:30** **Recap Yesterday && Questions**
- 09:30 - 10:15** **REST endpoint to modify and how to call them**
- 10:15 - 11:00** **Exercise “Add new users”**
- 11:00 - 11:15** **Coffee**
- 11:15 - 12:00** **REST endpoint to delete + Exercise**
- 12:00 - 12:45** **Lunch**

IDEMPOTENT

- What does it mean?
 - You can execute a call as many times as you want the outcome will be the same for the resource
- GET is idempotent
- DELETE is idempotent
 - First response will be 200 or 204, N+1 response will be a 404 or 500, but the resource is still deleted
- PUT is idempotent (same reason)
- POST is NOT idempotent

POST ENDPOINT

- Post endpoint is adding data to the resource (i.e. a row in the database)
- Important to create a new POJO to create the object (database id wise)
- Request body has to be added to the endpoint

WHAT IS A REQUEST BODY?

- JSON, XML and other serialisation forms
- JSON can be mapped to a Java object or any type of object

```
{  
  "name": "John",  
  "age": 12  
}
```

INSTALL POSTMAN

Postman is a client application to test REST endpoints

Demo

<https://www.postman.com/downloads/>

DAY 2 - AFTERNOON

12:45 - 13:00 What is the structure/components of a service?

13:00 - 13:45 Exercise: “Bring in the Structure in the UserService”

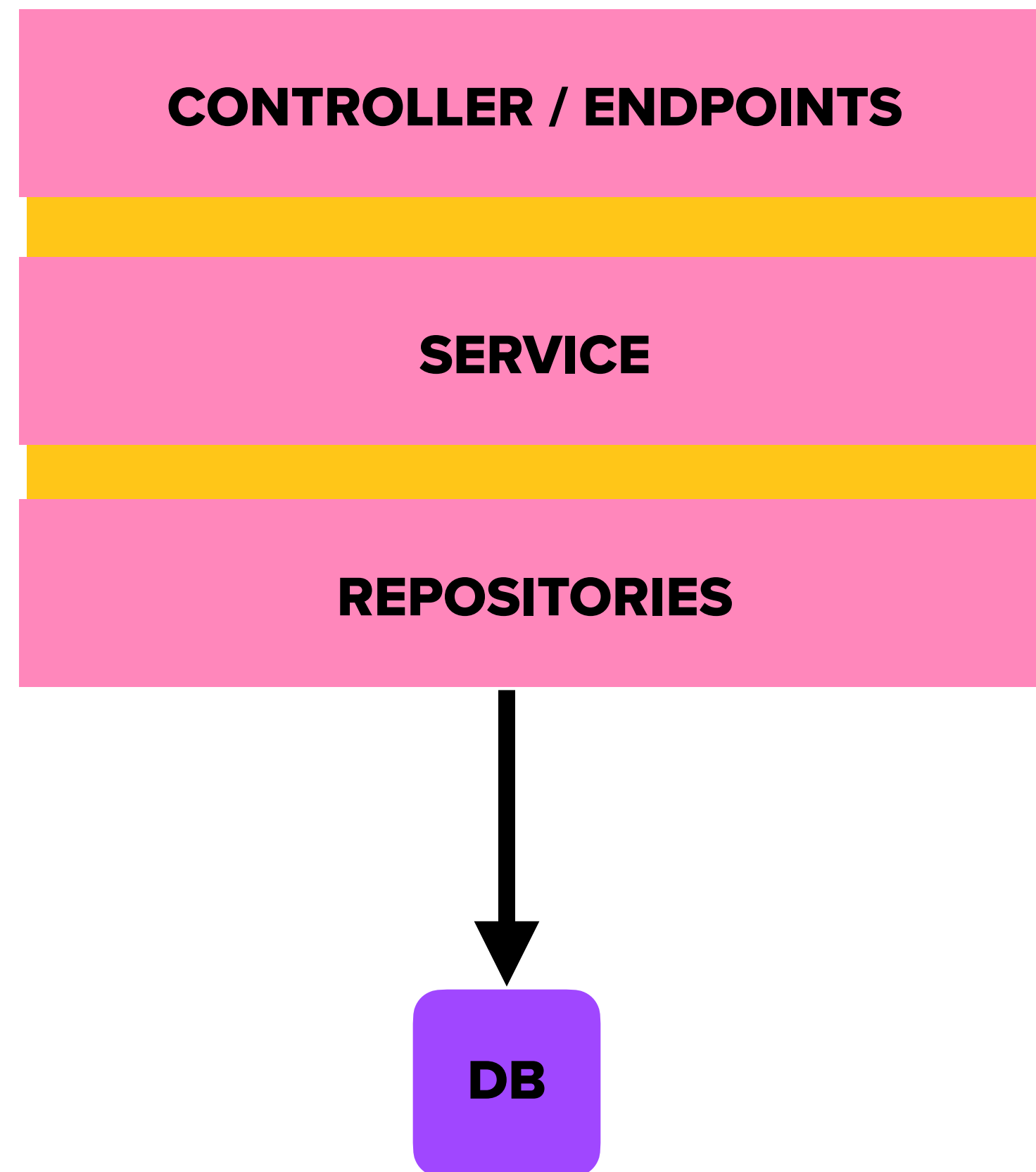
13:45 - 14:15 Microservices vs Monoliths

14:15 - 14:30 Coffee

14:30 - 15:00 Where can we find answers?

15:00 - 15:30 What can else can we do with Spring Boot

WHAT IS THE STRUCTURE OF A SPRING SERVICE



RESPONSIBILITIES PER LAYER

- **Controllers/Endpoints**
 - Define all the endpoints of an application (GET, POST, PUT, DELETE etc....)
 - Those should be not more than that
 - Keep them simple and clean
 - “Doorgeef-luikje”
- **Services**
 - Business logic is implement there
 - Objects are mapped to database entities and the other way around
- **Repositories**
 - Communicate with the database

DAO AND DTO

- **DAO**

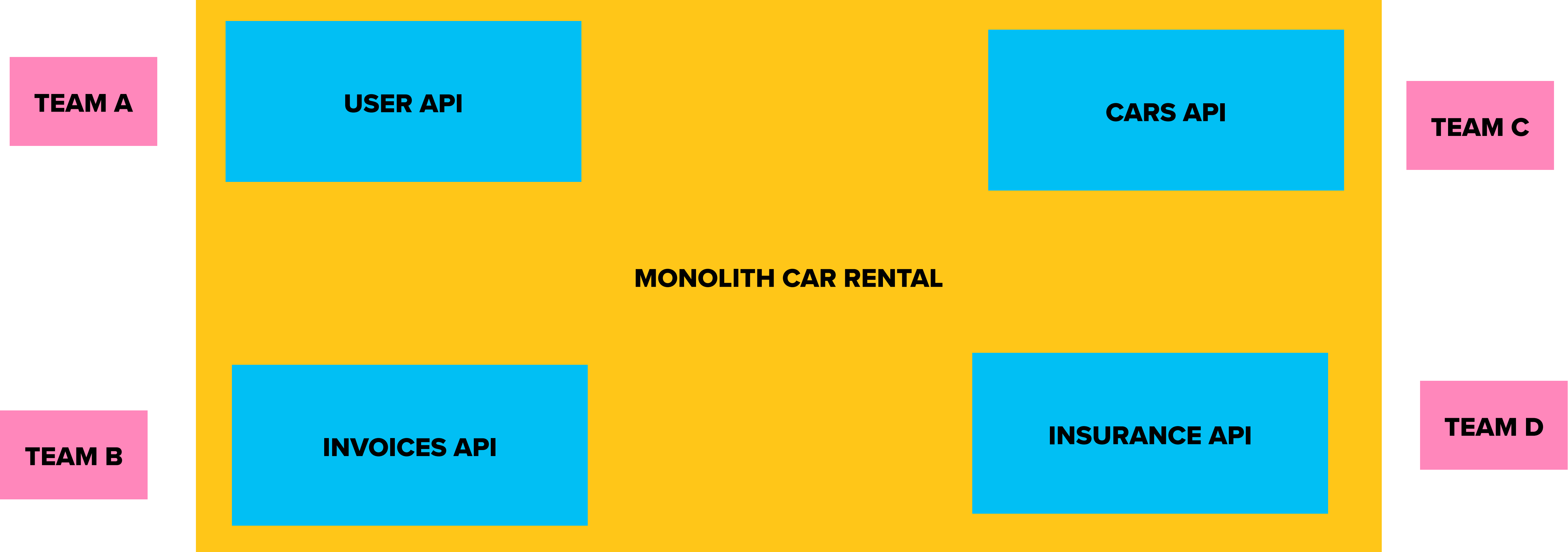
- **Data Access Object: most often a database entity, with an Id**

- **DTO**

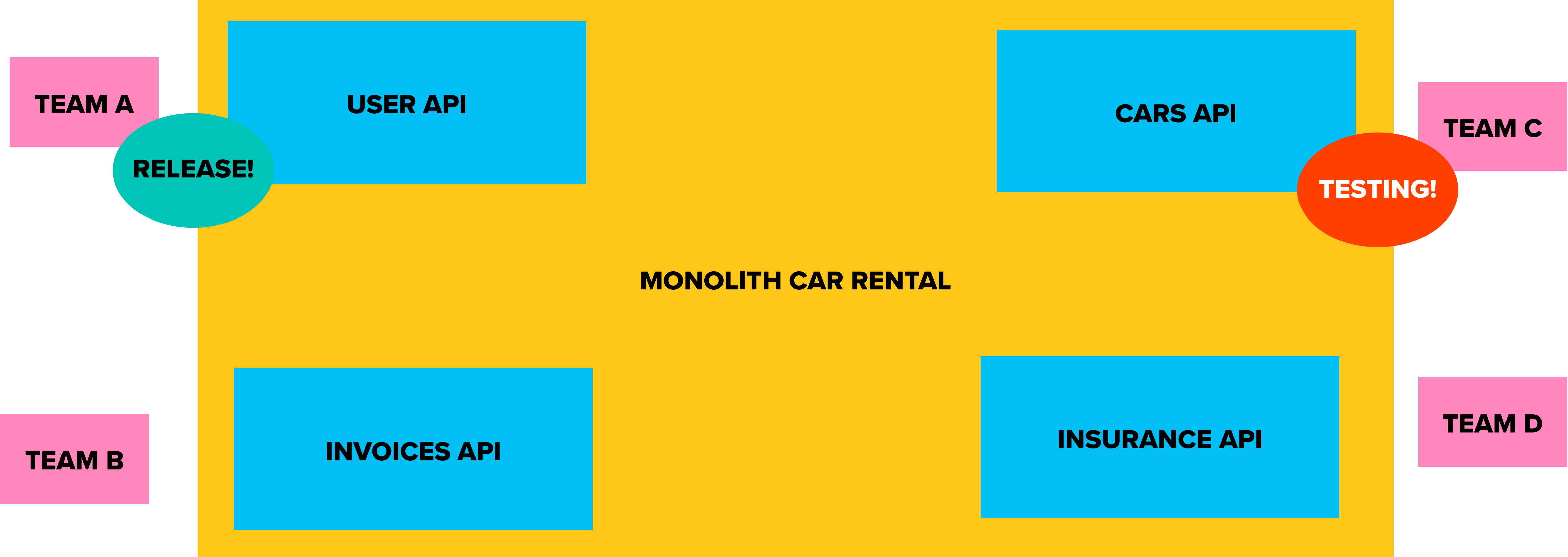
- **Data Transfer Object: serializable**

- **Be careful! DTO will and might not expose database properties (such as an id)**

MICROSERVICES VS MONOLITHS

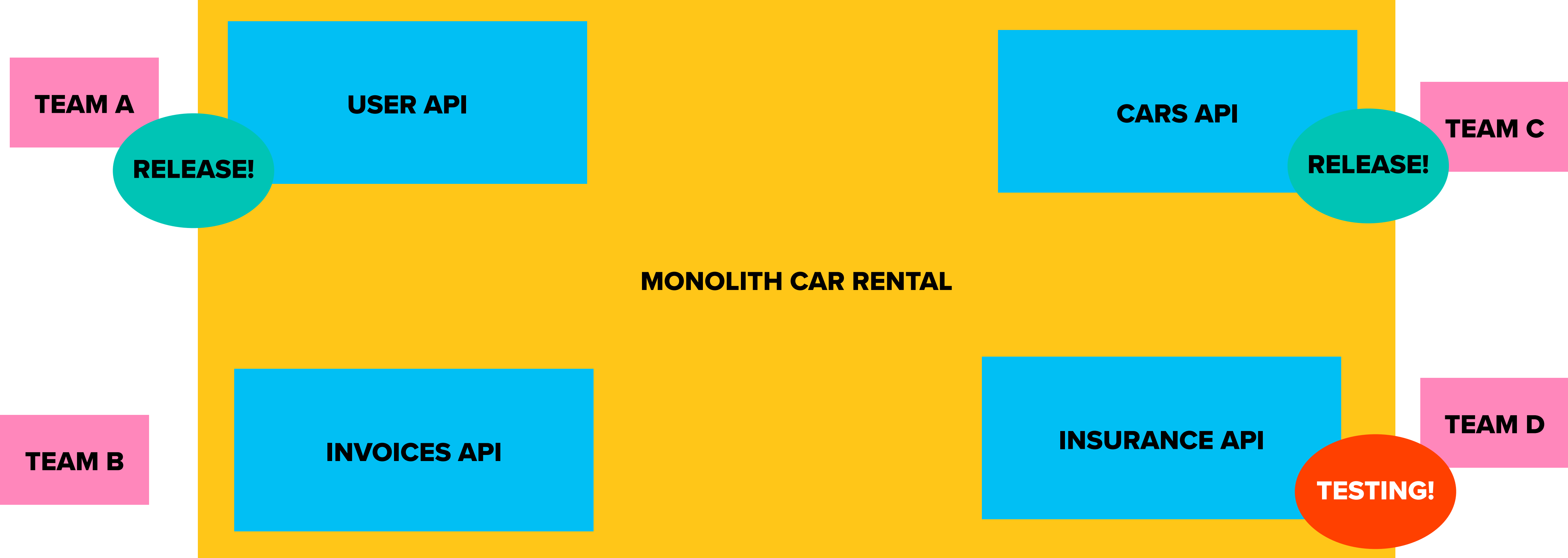


MONOLITIC RELEASES

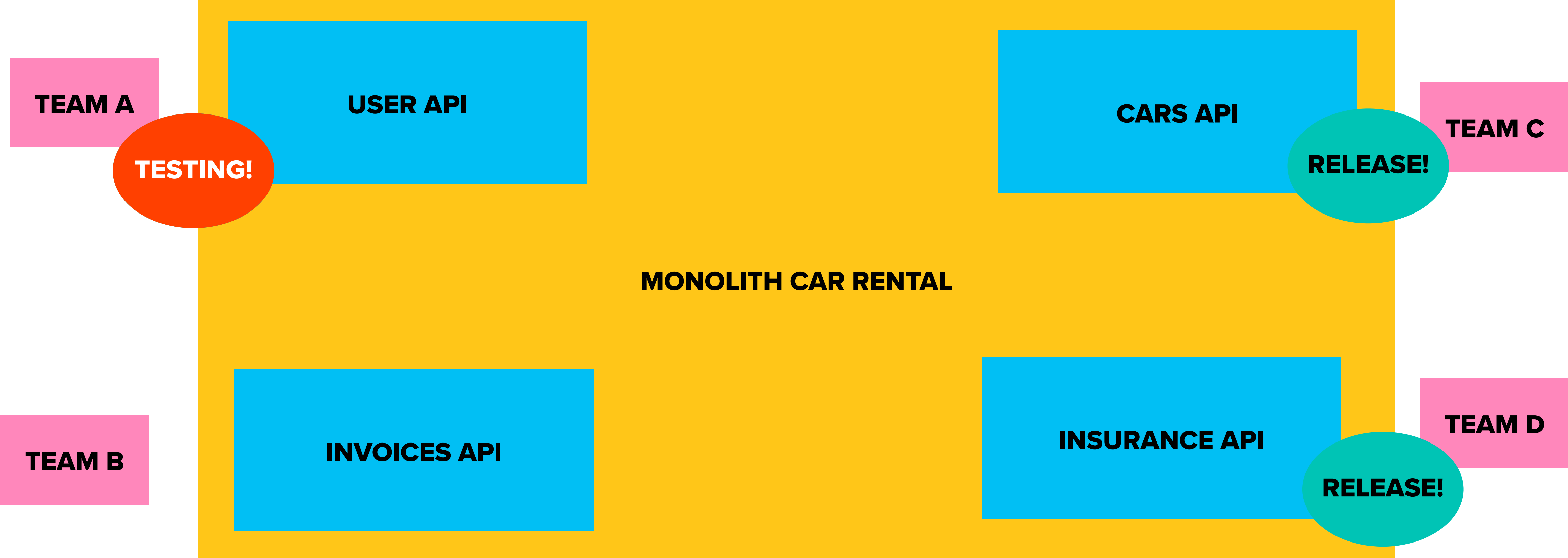


WHO WINS?

MONOLITIC RELEASES

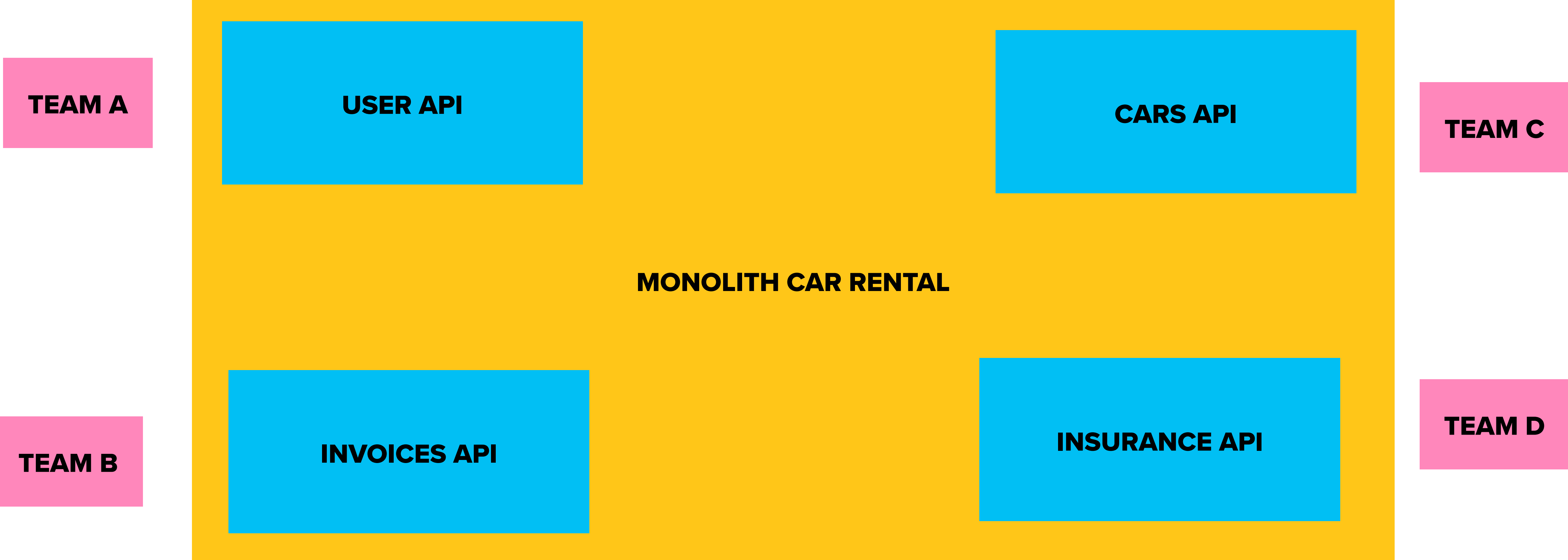


MONOLITIC RELEASES

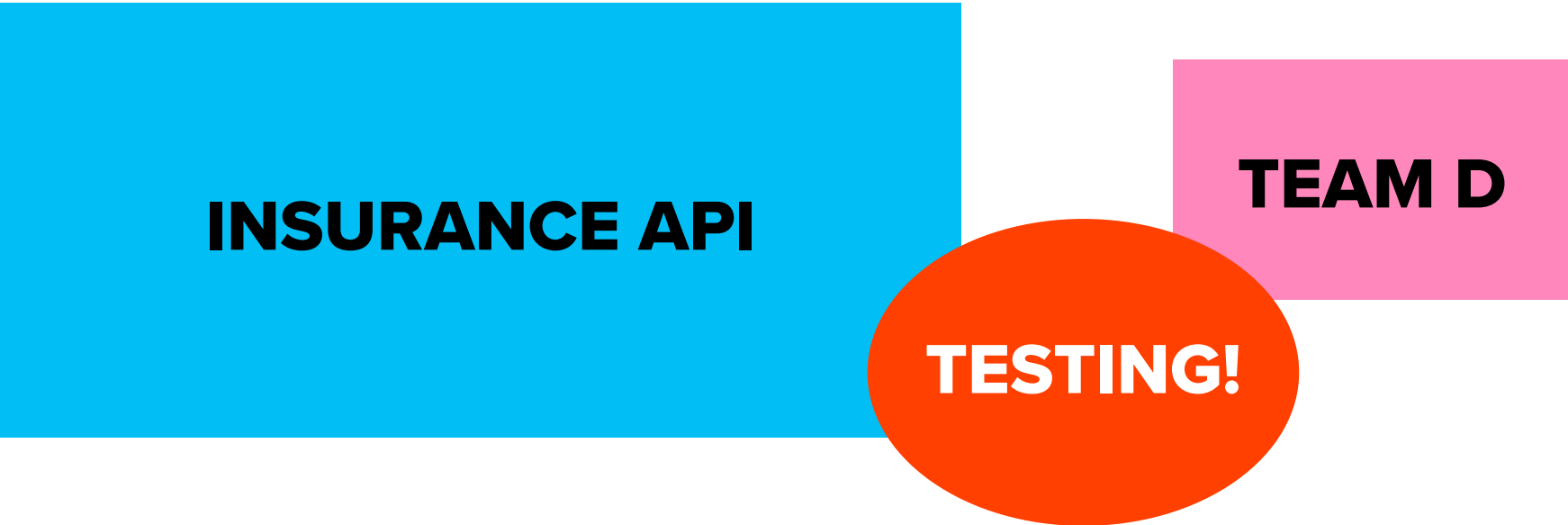
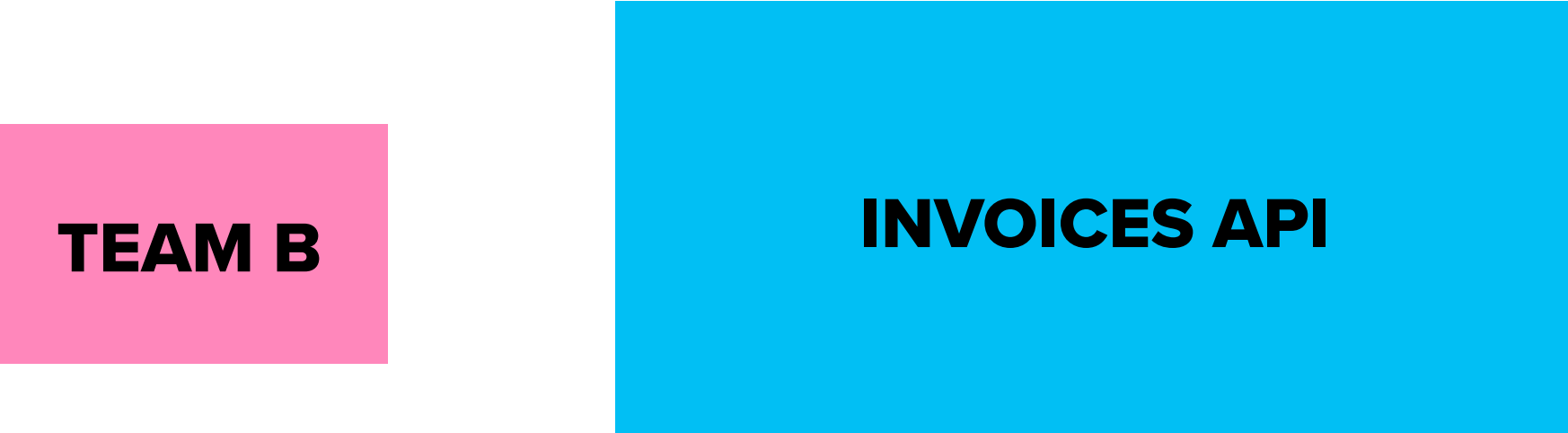


NEVER ENDING STORY!

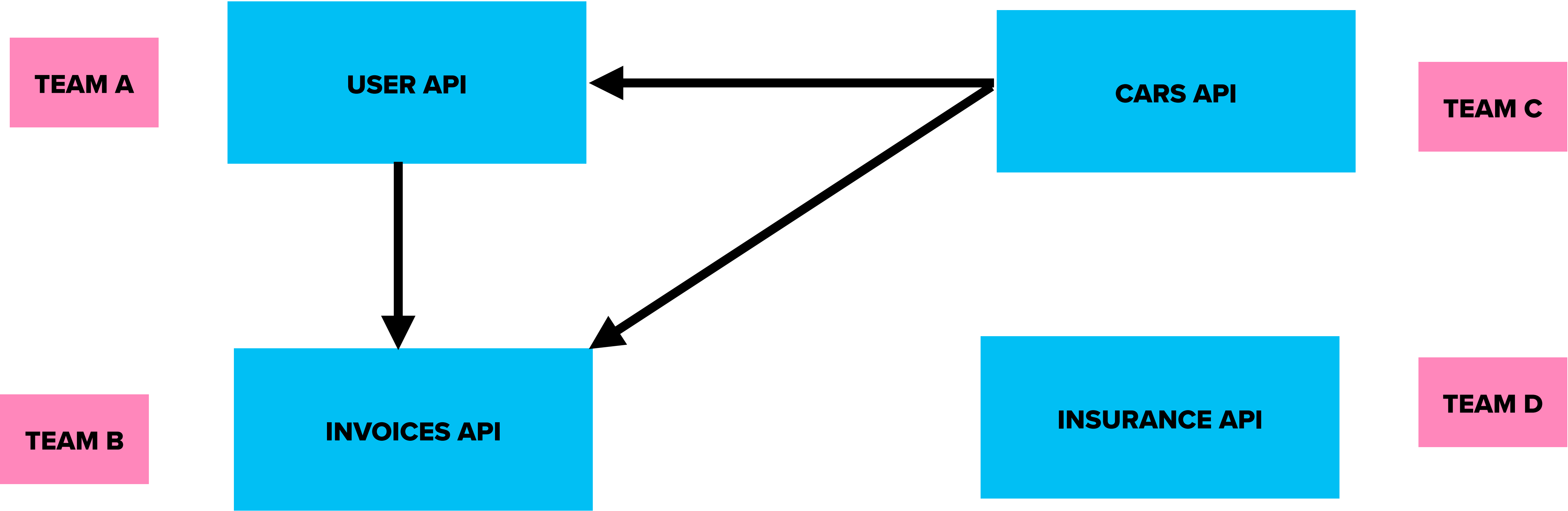
MICROSERVICES



MICROSERVICES



MICROSERVICES - CHALLENGES



**WHAT IS THE SCOPE OF A
MICROSERVICE?**

THANK YOU!