AAD / Backend Development

Testing

Erik van der Arend / Jan Willem Boer

23/24 Q3

SAXION
UNIVERSITY OF
APPLIED SCIENCES

Submission points

- Submit at the correct submission point:
  - DE @ the DE-submission point
  - BD @ the BD-submission point
  - HD @ the HD-submission point

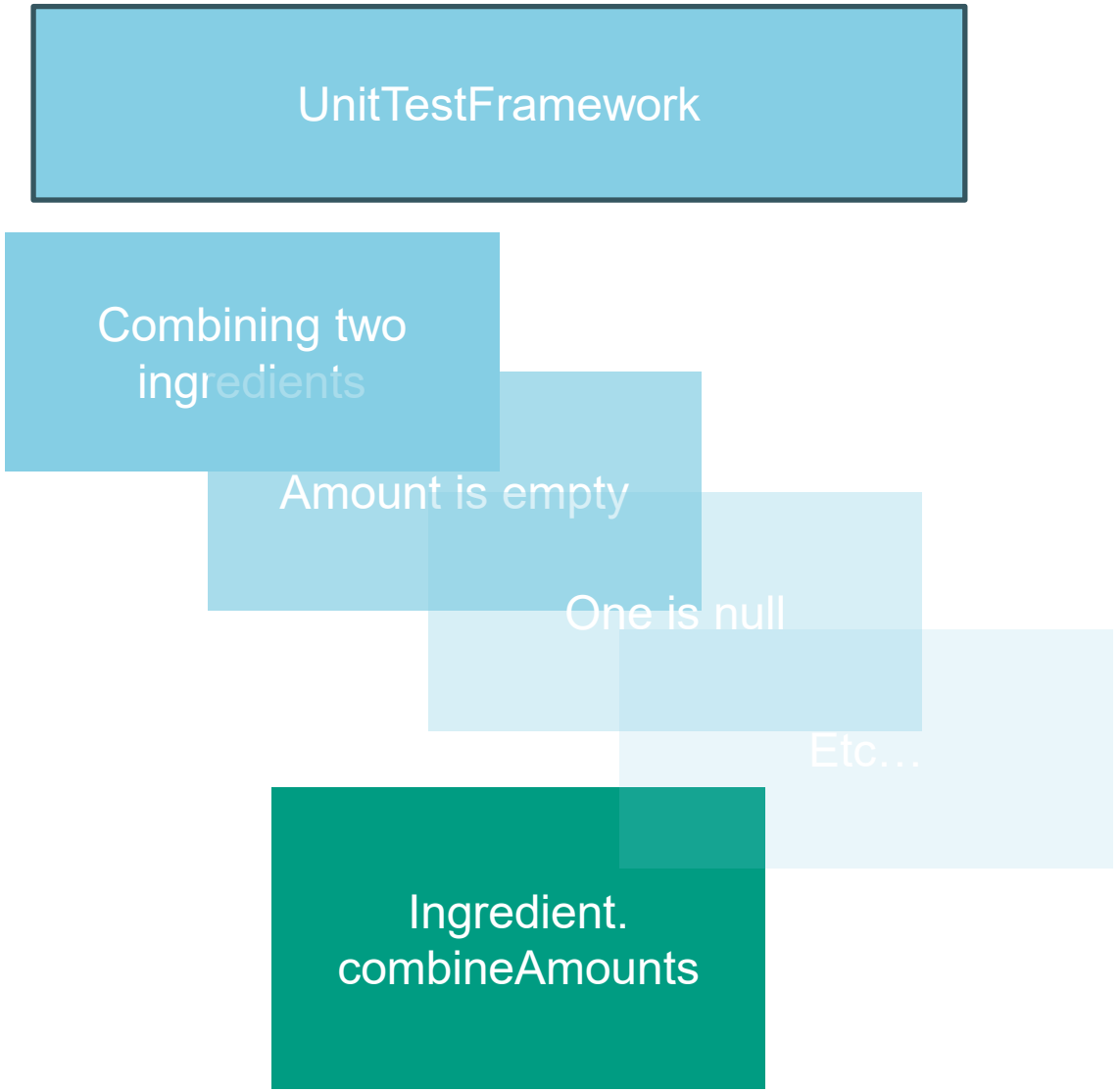
Look at the requirements in the assignment and advanced track documentation!
For example: there should be a show-and-tell document for the AT.

Testing

- Unit tests
- Application tests (API tests / UI tests)

Unit testing

- Testing one unit of code
- Minimizing dependencies



Naming and structure

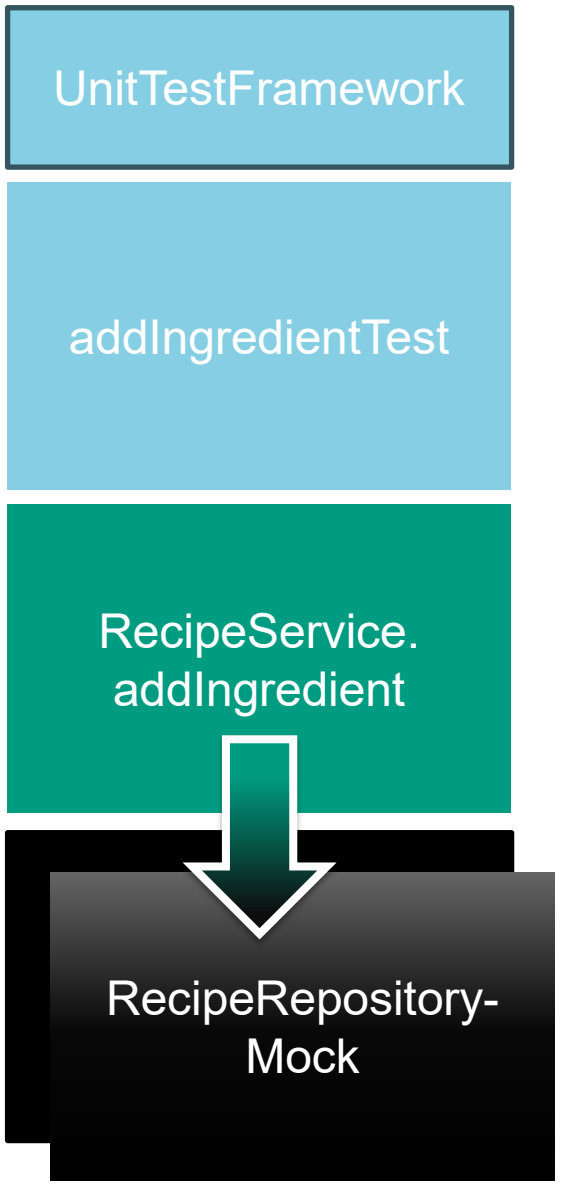
```java
@Test
public void CombineAmounts_adds_amount_to_first_ingredient() {
    // arrange
    Ingredient first = new Ingredient( name: "salt",  amount: "10 gr");
    Ingredient second = new Ingredient( name: "salt",  amount: "5 gr");

    // act
    first.combineAmounts(second);

    // assert
    assertEquals( expected: "15 gr", first.amount);
    assertEquals( expected: "Second should not be changed",  actual: "5 gr", second.amount);
}
```

Just describe the test in the name, but start with the unit under test (combineAmounts)

Use the Arrange / Act / Assert pattern in the test to structure your test

Unit testing

- Testing one unit of code
- Minimizing dependencies



Beware of testing the right thing

What should the test do, if the method looks like this?

```
public void addIngredient(Recipe recipe, Ingredient ingr) {
    this.repository.addIngredient(recipe.id, ingr);
}
```

On the mocked repo, check if addIngredient was called

Also check if the right arguments were passed

Beware of testing the right thing

Or this?

One test should test the situation where the ingredient is new and check if addIngredient was called and save was not called

Another test should add an existing ingredient and check if save was called and addIngredient was not called

```
public void addIngredient(Recipe recipe, Ingredient ingr) {
    // to prevent duplicates, see if the recipe doesn't
    // already have this ingredient
    var existingIngr = Arrays.stream(recipe.ingredients)
            .filter(x -> x.name.equalsIgnoreCase(ingr.name))
            .findFirst();

    if (existingIngr.isEmpty()) {
        // it is a new ingredient, add it.
        this.repository.addIngredient(recipe.id, ingr);
    } else {
        // it already has this ingredient, combine the amounts.
        existingIngr.get().combineAmounts(ingr);
        this.repository.save(recipe);
    }
}
```

Also check if combineAmounts was called, but that method should also have its own unit tests

New ingredient test

A mock version of the interface is created, only for this unit test

addIngredient arguments are checked

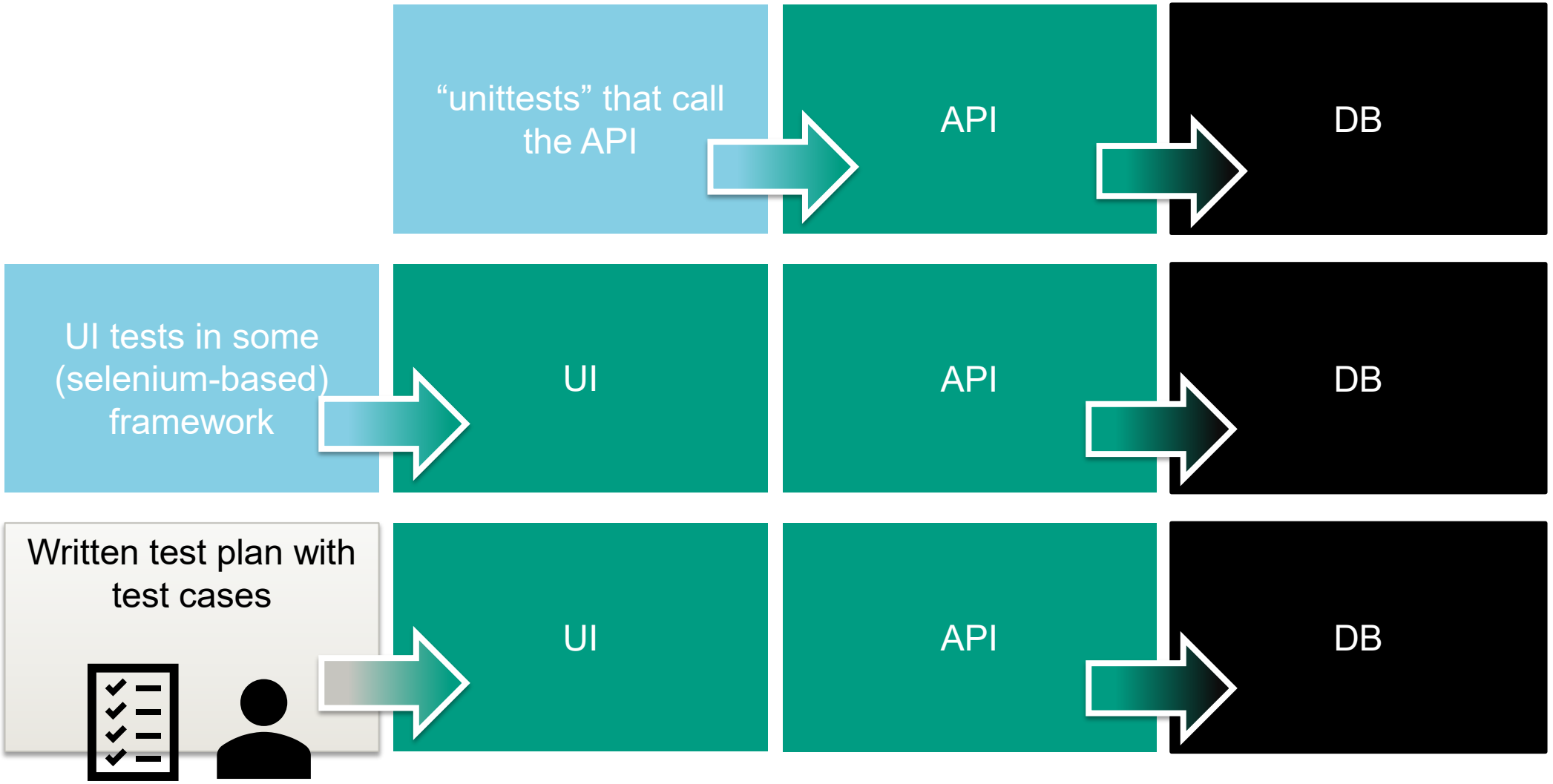
Fail if the wrong method is called

What did we fail to check in this test?

```
@Test
public void AddIngredient_adds_new_ingredient_to_repository() {
    // arrange
    var recipe = new Recipe( id: 1,  name: "Stamppot");
    var newIngredient = new Ingredient( name: "Boerenkool",  amount: "1 kg");
    // create the service under test with a mock repository
    var is = new IngredientsService(new RecipeRepository() {
        1 usage
        @Override
        public void addIngredient(int recipeId, Ingredient ingr) {
            assertEquals( expected: 1, recipeId,  message: "RecipeID incorrect");
            assertEquals(newIngredient, ingr,  message: "Ingredient incorrect");
        }
        1 usage
        @Override
        public void save(Recipe recipe) {
            fail("Save should not be called on a new ingredient");
        }
    });

    // act
    is.addIngredient(recipe, newIngredient);

    // (asserts are in the repository mock)
}
```

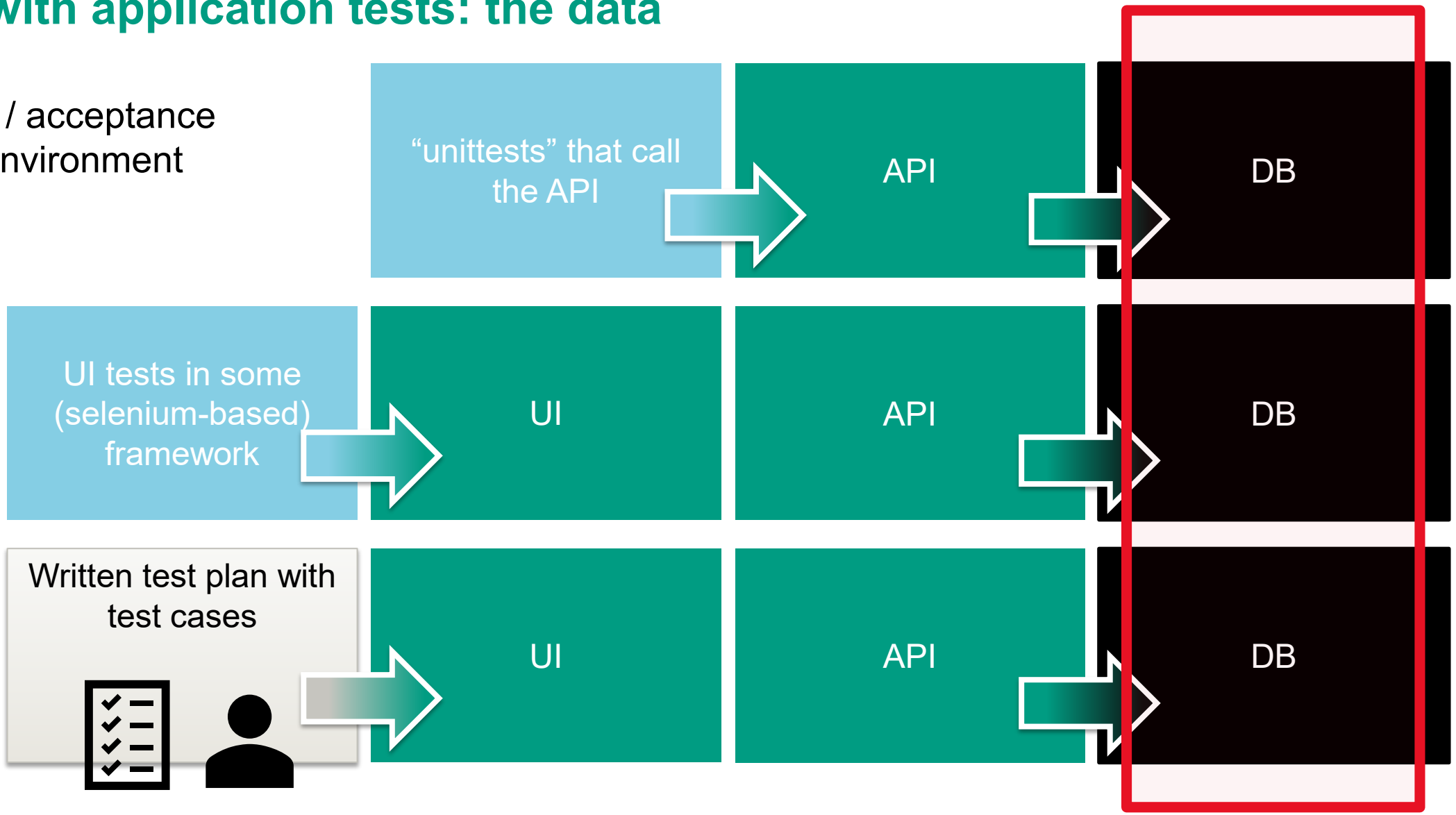
Application tests

- API
- UI
- Manual



Problem with application tests: the data

- Use a test / acceptance / staging environment

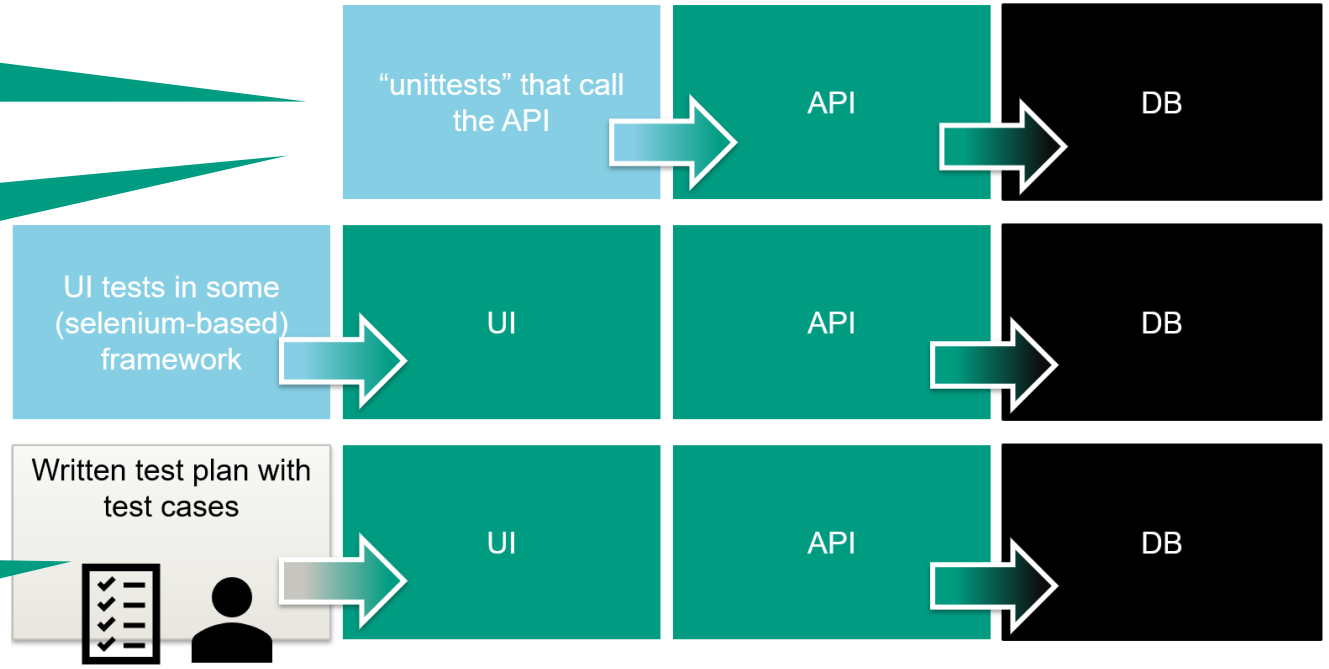


Data problems: repeatability

An API tests add a new ingredient to a recipe and checks the ingredient is added

But the second time this test is done, the ingredient is not added, but the amounts will be combined

Or: one of the testers removed the recipe entirely “to make sure something worked”.



Making tests repeatable

Solution is easy but tedious

- Before each test:
  - Drop the database
  - Create a new database
  - Seed with test data

- Perform test

- Check results

The same goes for other data (files, external data)

How is this relevant

Have tests for the important functionality

Have an extensive coverage in case of the QM track

Now what

Finish and submit your assignment 1 and 2 before Monday 9:00

Next week

- Prepare for the **assessment** (see first slide previous week)
- Start with your **project group** and contact the client for the project