ط

هل الـ Unit Test فعلًا مهم؟!

ولماذا نختبر دوال وأمور نحن كتبناها ؟
ومتأكدون تمامًا منها ومما تفعله !

Ahmed El-Tabarani

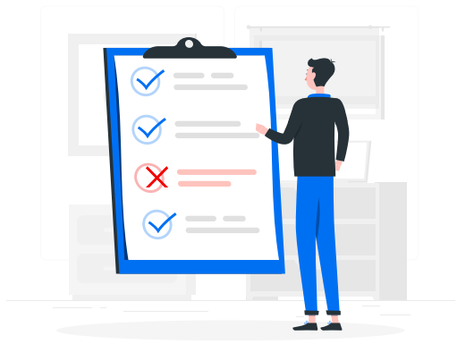
Back-End Developer

01

لماذا نختبر ؟

عندما سمعت عن الـ unit test لأول مرة قلت لماذا أختبر دالة كتبتها بنفسي وأعلم نتيجتها ؟

الأمر بسيط، نحن نختبر لأنه في حالة قام شخص بتعديل الدالة مستقبلًا سيقوم الـ unit test بتنبيهه بأنه قام بتعديلات في الدالة تخل بالنواتج التي كنا نتوقعها منها



صورة نادرة للـ unit test أثناء تأكده من النواتج المتوقعه من الدالة

دعونا نرى الأمر بشكل عملي

Ahmed El-Tabarani
Back-End Developer

لننشيء دالة ونفكر!

أنت قمت بعمل دالة getUserById تقوم بإرجاع مستخدم معين من قاعدة البيانات بناءً على الـ id

```ts
async function getUserById(id: string): Promise<User> {
  const user = await userModel.findById(id);

  if (!user)
    throw new NotFoundException("User doesn't exist");

  delete user.password;
  return user;
}
```

هنا علينا طرح سؤالين مهمين:

- ما الذي تتوقعه من الدالة؟
  - إحضار بيانات المستخدم بشكل معين
  - عرض خطأ بأن المستخدم غير موجود
- هل تعتمد على أمور خارجية؟
  - تعتمد على قاعدة البيانات من أجل البحث

Ahmed El-Tabarani
Back-End Developer

محاكاة البيانات وتزيفها قبل الاختبار

لاحظ أن الدالة تعتمد على قاعدة بيانات حقيقية

لذا علينا محاكاتها قبل ان نختبر الدالة لكي لا

نتسبب بأي ضرر لقاعدة البيانات الخاصة بنا

لذا سنقوم بعمل محاكاة للـ userModel ودالة

findById الخاصة بها

وسنقوم أيضًا بإنشاء دالة تقوم بإرجاع مستخدم

بالشكل المطابق للـ schema الخاصة به

ملحوظة: عملية المحاكاة وتزييف الأشياء تسمى Mocking

ونحن نقوم بعمل محاكاة بهذا الشكل:

`getUserById()` ⟵ `mockUserModel.findById` ⟵ `getMockUser()`

يستدعي قاعدة بيانات مزيفة — يرجع بيانات مستخدم مزيف

Ahmed El-Tabarani
Back-End Developer

الغرض من عملية الـ Mocking

لنفترض أنك تختبر الطباخ الذي يعمل لديك
وهو يعتمد على بعض الأدوات، أنت لا تريد أن
تختبر الأدوات بل تريد أن تختبر الطباخ

هنا سنقوم بمحاكاة الأدوات ونختبر الحالات

- إذا توفرت كل الأدوات

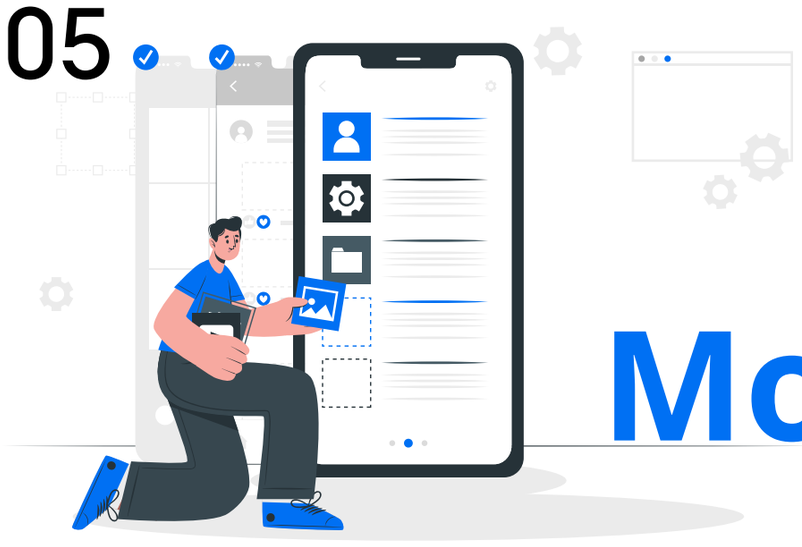
هل سينجز الطباخ عمله بشكل صحيح وكما هو متوقع ؟

- إذا لم تتوفر كل الأدوات

هل سيقوم الطباخ بالتبليغ عن عدم توفرها كما هو متوقع أم سيطبخ طبخة سيئة وناقصة المكونات ؟

Ahmed El-Tabarani
Back-End Developer

ط



تطبيق مفهوم الـ Mocking

لنقم بعمل دالة تحاكي بيانات المستخدم التي في قاعدة البيانات :

```
const getMockUser = (): User => {
  return {
    id: 'user-id-123',
    fullName: 'user-name',
    email: 'user_email@domain.com',
    password: '1234567890',
  }
};
```

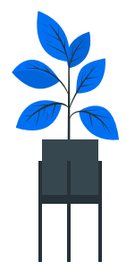
ملحوظة: نحن نضع قيم افتراضية لأننا نريد أن نختبر الشكل العام وليس القيم بحد ذاتها

لنحاكي الآن الـ userModel ونزيف الناتج الراجع من الدالة findById بمستخدم مزيف:

```
const mockUserModel = {
  findById: jest.fn().mockResolvedValue(getMockUser()),
};
```

الآن نستخدم دالة spyOn لنخبر Jest أن يستبدل الدالة الحقيقية userModel.findById بالمزيفة mockUserModel.findById

```
jest.spyOn(userModel, 'findById')
    .mockImplementation(mockUserModel.findById);
```



Ahmed El-Tabarani
Back-End Developer

ط

لنحلل ما سوف نختبره

ما الحالات المختلفة التي يمكن أن تحدث

للدالة التي سنختبرها getUserById ؟

- إذا أرجعت الـ userModel.findById بيانات المستخدم من قاعدة البيانات بشكل صحيح

  هل ستقوم الدالة getUserById بإرجاع نفس البيانات ؟

  لأنه قد يقوم شخص بجعل الدالة لا ترجعها أو تقوم بحذف بعض البيانات

- إذا لم ترجع الـ userModel.findById بيانات المستخدم بشكل صحيح

  هل ستقوم الدالة بإرجاع NotFoundException ؟

  لأنه قد يقوم شخص بجعل الدالة تقوم بإرجاع null أو حتى BadRequestException أو أي شيء آخر غير الذي نريده

كل هذه الأمور يجب أن نفكر بها قبل أن نبدأ

بعمل الـ unit test للدالة

Ahmed El-Tabarani
Back-End Developer

وصلنا للجزء الممتع

بعدما انتهينا من الـ mocking ومعرفة الحالات التي سنختبرها نأتي لأمتع جزء وهو الاختبار الفعلي

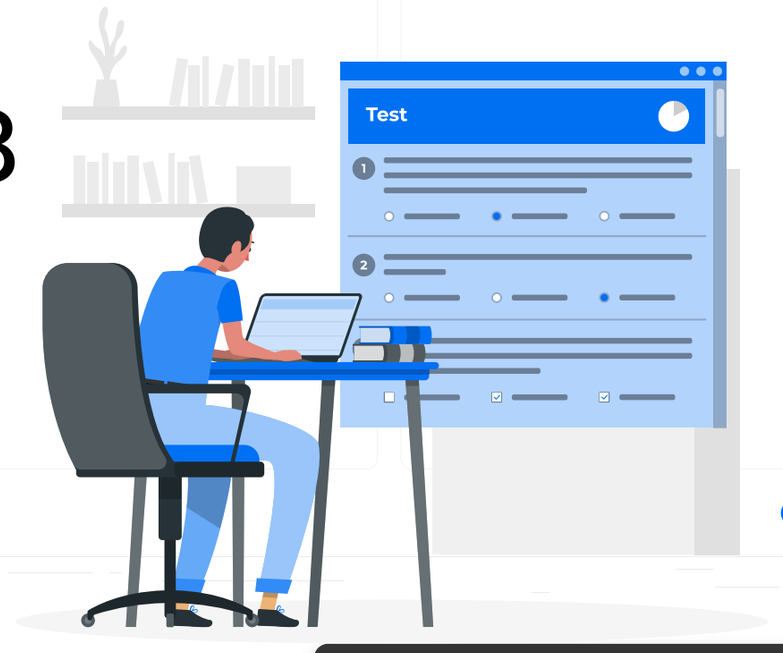
لنتذكر الدالة التي نريد اختبارها getUserById

```ts
async function getUserById(id: string): Promise<User> {
  const user = await userModel.findById(id);

  if (!user)
    throw new NotFoundException("User doesn't exist");

  delete user.password;
  return user;
}
```

ولنتذكر أننا نريد أن نختبر الدالة في حالتين:

- هل ترجع بيانات المستخدم بشكل سليم ان كان موجود
- هل تقوم بعمل NotFoundException إن كان غير موجود

Ahmed El-Tabarani
Back-End Developer



لنبدأ اختبار الدالة الآن!

```
test('it should get a user by id', async () => {
  const expectedUser = getMockUser();
  // we expect that getUserById will omit the password
  delete expectedUser.password;

  // call the function to test its behavior
  const result = await getUserById(getMockUser().id);

  // test the result
  // should return the same result that we expect
  expect(result).toEqual(expectedUser);
});
```

استخدمنا دالة test لبدأ الاختبار وقمنا بكتابة وصف توضيحي لهذا الاختبار
ثم عرفنا شكل البيانات التي نتوقعه ان يرجع وستلاحظ أننا لا نريد ان تقوم الدالة بإرجاع الـ password
وفي حالة أن الدالة أرجعته فسيقوم الـ jest بإخبارنا أن الاختبار فشل

```
 PASS  ./user.test.ts
  √ it should get a user by id (2 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        5.362 s
Ran all test suites.
```

الآن الـ jest سيقوم بالتحقق من أن الدالة ترجع لنا الناتج الذي نتوقعه أم لا وعندما تفعل يعطينا رسالة جميلة تدل على النجاح كما ترى

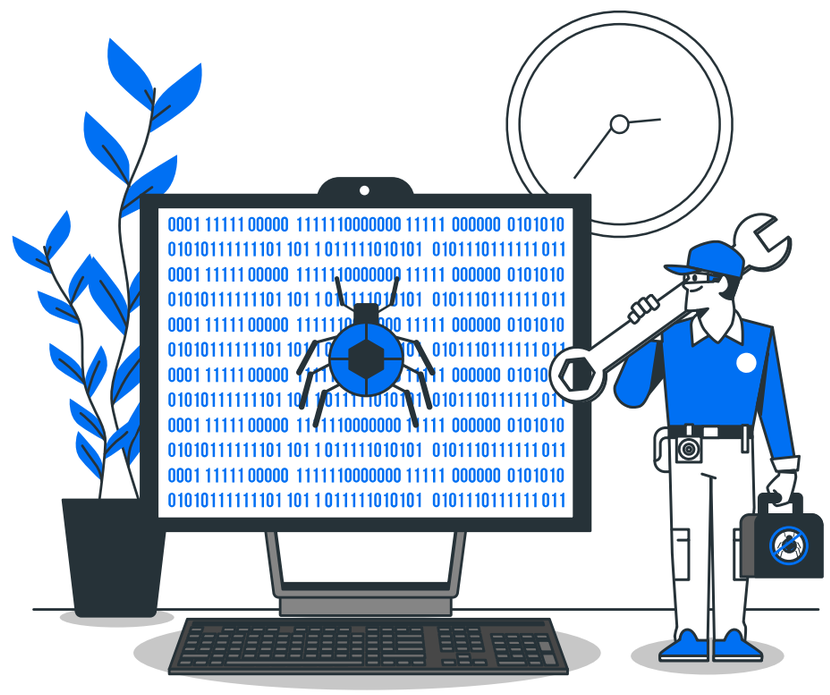
Ahmed El-Tabarani
Back-End Developer

لماذا نختبر إن كان سينجح؟

حسنًا بعد ما اختبرنا أول حالة، لنتخيل أن زميلك في العمل قام بتعديلات ومنها انه جعل دالة getUserById ترجع password

```
FAIL  ./user.test.ts
 × it should get a user by id (5 ms)

 ● it should get a user by id

   expect(received).toEqual(expected) // deep equality

   - Expected  - 0
   + Received  + 1

     Object {
       "email": "user_email@domain.com",
       "fullName": "user-name",
       "id": "user-id-123",
   +   "password": "1234567890",
     }

      51 |   // test the result
      52 |   // should return the same result that we expect
    > 53 |   expect(result).toEqual(expectedUser);
         |                  ^
      54 | });
      55 |

      at Object.<anonymous> (user.test.ts:53:18)
      at fulfilled (user.test.ts:4:58)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        4.845 s
Ran all test suites.
```

ثم قام بتشغيل الاختبارات ليتأكد أنه لم يفسد شيء ليجد أن الـ jest ينبهه بأن هناك خطأ وسيجد رسالة تخبره بالتفاصيل



صورة لزميلك بعد إصلاحه للخطأ، وهو سعيد لأن الـ jest نبهه لها قبل ان يرفع على production

ملحوظة: كان سيخسر وظيفته ان لم ينبه الـ jest

Ahmed El-Tabarani

Back-End Developer

اختبار الدالة عند عدم وجود مستخدم

هنا سنقوم بعمل تعديل للـ findById ونجعلها ترجع null لانها حاليا ترجع لنا mockUser كما عرفناها في الـ mockUserModel

لكن الآن نريد أن نتأكد من أن الدالة تقوم بإرجاع NotFoundException في حالة عدم وجود المستخدم

لنقوم بذلك سنستخدم jest.spyOn مجددًا لتغير الـ implementation اثناء اختبارنا

Ahmed El-Tabarani
Back-End Developer

اختبار الحالة الثانية

```
test('should throw a NotFoundException if user not found', async () => {
  jest.spyOn(userModel, 'findById').mockResolvedValueOnce(null);

  await expect(getUserById(getMockUser().id)).rejects.toThrow(
    NotFoundException
  );
});
```

قمنا بعمل mocking جديد للـ findById وجعلناها ترجع null لنحاكي الحالة ولاحظ استخدمنا mockResolvedValueOnce لأننا نريد بعد اختبار تلك الحالة أن نعود للـ implementation الافتراضي

بغض النظر عن التفاصيل الآن نحن نختبر إذا كانت الدالة فعلًا ترجع NotFoundException أم لا وأصبح لدينا unit test

```
 PASS  ./user.test.ts
  √ it should get a user by id (3 ms)
  √ should throw a NotFoundException if user not found (5 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.666 s, estimated 2 s
Ran all test suites.
```

Ahmed El-Tabarani

Back-End Developer

أظن أن أحدهم غير في الدالة مجددًا!

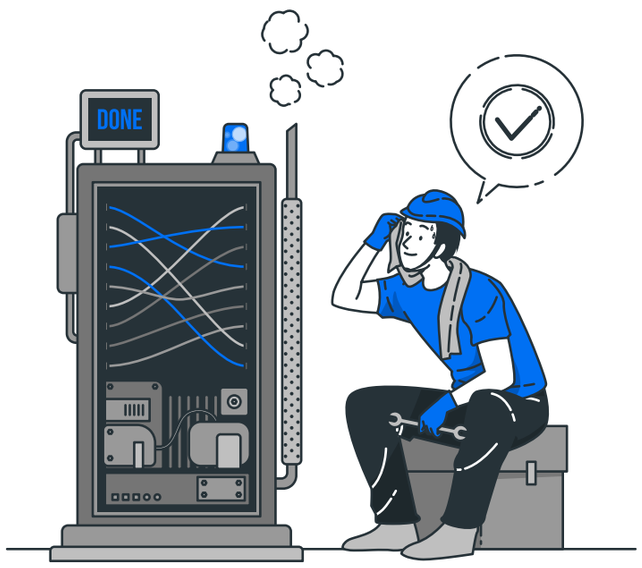
```
FAIL  ./user.test.ts
 √ it should get a user by id (3 ms)
 × should throw a NotFoundException if user not found (23 ms)

 ● should throw a NotFoundException if user not found

   expect(received).rejects.toThrow(expected)

   Expected constructor: NotFoundException
   Received constructor: BadRequestException

   Received message: "User doesn't exist"

       43 |   const user = await userModel.findById(id);
       44 |
     > 45 |   if (!user) throw new BadRequestException("User doesn't exist");
          |                      ^
       46 |
       47 |     delete user.password;
       48 |   return user;

       at user.test.ts:45:20
       at fulfilled (user.test.ts:4:58)

   67 |   jest.spyOn(userModel, 'findById').mockResolvedValueOnce(null);
   68 |
 > 69 |   await expect(getUserById(getMockUser().id)).rejects.toThrow(
      |                                                       ^
   70 |     NotFoundException
   71 |   );
   72 |   expect(mockUserModel.findById).toHaveBeenCalledWith(getMockUser().id);

       at Object.toThrow (node_modules/expect/build/index.js:218:22)
       at Object.<anonymous> (user.test.ts:69:55)
       at user.test.ts:7:71
       at Object.<anonymous>.__awaiter (user.test.ts:3:12)
       at Object.<anonymous> (user.test.ts:66:71)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 passed, 2 total
Snapshots:   0 total
Time:        0.687 s, estimated 1 s
Ran all test suites.
```

Ahmed El-Tabarani

Back-End Developer

الخطوات التي سنتبعها لاختبار أي دالة

- نعرف البيانات التي سنرسلها للدالة
  في حالة إذا كانت الدالة تستقبل بيانات
- نعرف شكل البيانات التي نتوقعها أن ترجع من الدالة
- نقوم بعمل mocking لأي دالة او مكتبة خارجية ليست ضمن ما نريده اختباره
  - مثل ما قمنا مع mockUserModel و getMockUser
  - أو نستخدم jest.spyOn أثناء الاختبار في حالة اذا أردت تغير الـ implementation الافتراضي الذي وضعناه في mockUserModel
- نستدعي الدالة التي نريد اختبارها
- نختبر اذ ا كانت ترجع لنا نفس النتيجة التي نتوقعها دون تغير
- نختبر الأمور الجانبية الأخرى



Ahmed El-Tabarani
Back-End Developer

أنا أحمد الطبراني، مهندس برمجيات SWE 😁

مبرمج متخصص في عالم الـ Backend ⚙️💻

أحب دائمًا أن أُشارك معرفتي المتواضعة مع الآخرين لعله عسى أن يستفيد شخص ما بما أكتبه 🤗
أكتب بعض المقالات عن أشياء مختلفة في عالم البرمجة، أرجوا أن تستفيدوا وتستمتعوا 😉

تابعني علي لينكدإن او علي مدونتي الشخصية