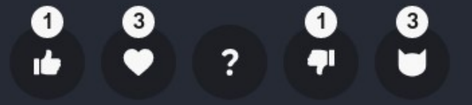
Comment TU peux laisser un monde meilleur à tes enfants ?

Instructions

Laurent Babin

- 32 ans
- Capgeminien depuis 2013
- Mairie de Monaco, ETSI, Amadeus, Kone, GSF
- .NET / Angular




DESIGNED IN THE FRENCH ALPS


Mentimeter
1
3

Je lègue

ou pas

Une maison

Je lègue

17

ou pas

3

Des dettes

Je lègue

6

ou pas

17

Des valeurs

Je lègue

23

ou pas

1

Des maladies

Je lègue

7

ou pas

16

Un Test Unitaire comme ça

```
describe("test", () => {
    it("test", () => {
        var trueValue = true;
        var falseValue = false;
        let baby = new Baby();
        baby.isTired = true;
        jest.spyOn(TimeService, 'currentDate', 'get')
            .mockReturnValue(moment(21, 'HH'));
        expect(baby.needToSleep).toBe(true);
        expect(baby.isTired).toBe(trueValue);
        expect(baby.isDiaperFull).toBe(falseValue);
        expect(baby.isHungry).toBe(falseValue);
    });
});
```

Je lègue

2

ou pas

9

Mentimeter


Mentimeter
NO
1

Comment TU peux laisser un monde meilleur à tes enfants ?

Test Unitaire (aka: TU, unit test, UT, ...)

procédure permettant de vérifier le fonctionnement attendu d'une partie précise d'un logiciel

```ts
export function add(x: number, y: number): number {
  return x + y
}
```

```
it("should return 15 when add 5 to 10", () => {
    const result = add(10, 5);

    expect(result).toBe(15);
});
```

Mentimeter

A quoi ça sert ?

2 16

1. Permet de tester rapidement son code

Efficacité

Mentimeter

2. Réduit les risques de régressions

Qualité

3. Documente notre code

Valeur documentative

4. Tranquillise toutes les parties prenantes

Confiance

5. 90 % de coverage : fait beau dans une présentation client

```ts
// --- math.service.ts
export function divide(x: number, y: number): number {
  return x / y;
}

// --- math.service.spec.ts
it("should return 2 when divide 4 by 2", () => {
  const result = divide(4, 2);

  expect(result).toBe(2);
}); //SUCCESS
```

```
// --- planning.service.ts
get isNight(): boolean {
  return AstroService.isEclipse || TimeService.currentDate.isBetween(
      TimeService.getTime(20),
      TimeService.getTime(6).add(1, 'day'));
}
// --- planning.service.spec.ts
beforeEach(() => {
  jest.spyOn(AstroService, 'isEclipse', 'get').mockReturnValue(true);
})
it("should be night when it's 21:00", () => {
  mockCurrentHour(21);
  const isNight = planningService.isNight;
  expect(isNight).toBeTruthy();
}); //SUCCESS
```

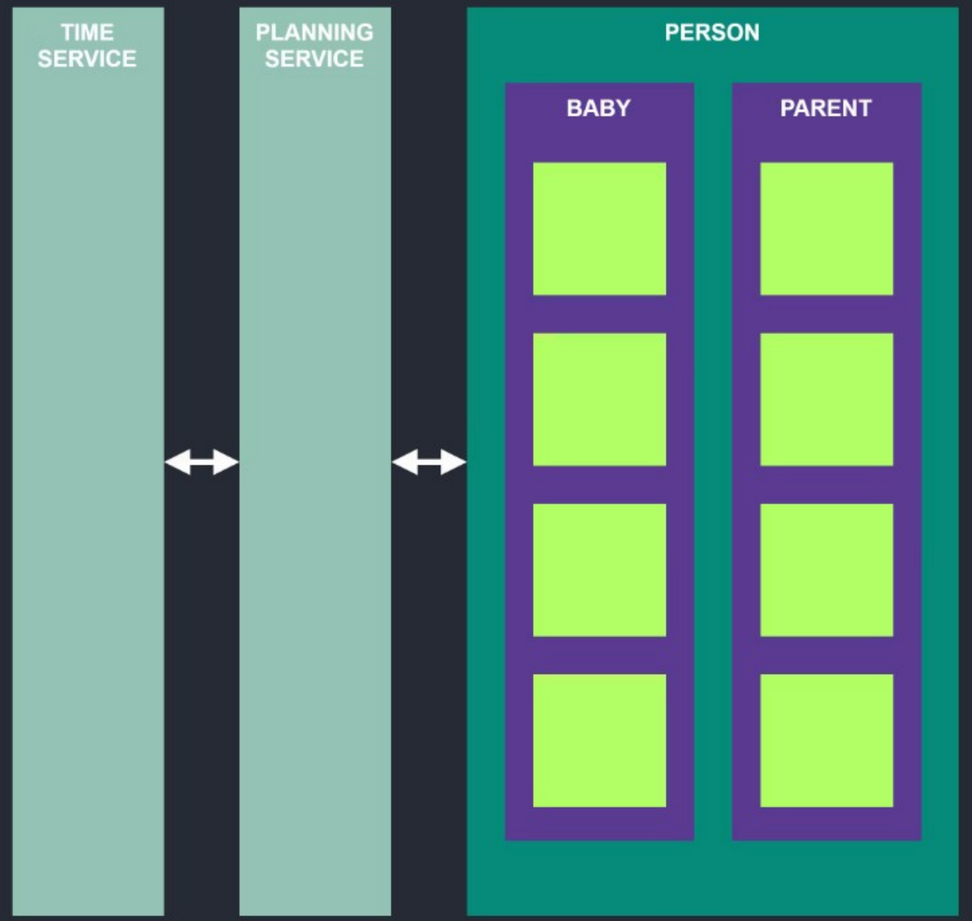
3 piliers des tests unitaires

SMS

- Structure
- Mention
- Scope

```
export default abstract class Person {
  //...

  public isTired: boolean = false;

  get needToSleep(): boolean {
    return this.planningService.isNight && this.isTired;
  }
}
```

```
export default class PlanningService {

    get isNight(): boolean {
        return AstroService.isEclipse || TimeService.currentDate.isBetween(
            TimeService.getTime(20),
            TimeService.getTime(6).add(1, 'day'));
    }

}
```



```
describe("test", () => {
    it("test", () => {
        var trueValue = true;
        var falseValue = false;
        let baby = new Baby();
        baby.isTired = true;
        jest.spyOn(TimeService, 'currentDate', 'get')
            .mockReturnValue(moment(19, 'HH'));
        expect(baby.needToSleep).toBe(falseValue);
        expect(baby.isTired).toBe(trueValue);
    });
});
```

Mentimeter

```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
}

describe("needToSleep", () => {
    it("should not need to sleep when it's day time even if tired", () => {
        jest.spyOn(PlanningService.prototype, 'isNight', 'get').mockImplementation(() => false);
        baby.isTired = false;

        const result = baby.needToSleep;

        expect(result).toBeFalsy();
    });
});
```

Structure

Les 3As

- Arrange
- Act
- Assert

Mentimeter

```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
}

describe("needToSleep", () => {
    it("should not need to sleep when it's day time even if tired", () => {
        // Arrange
        jest.spyOn(PlanningService.prototype, 'isNight', 'get').mockImplementation(() => false);
        baby.isTired = false;

        // Act
        const result = baby.needToSleep;

        // Assert
        expect(result).toBeFalsy();
    });
});
```

Pilier #1 - Structure - Arrange

1

```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
}

describe("needToSleep", () => {
    it("should not need to sleep when it's day time even if tired", () => {
        jest.spyOn(PlanningService.prototype, 'isNight', 'get').mockImplementation(() => false);
        baby.isTired = false;

        const result = baby.needToSleep;

        expect(result).toBeFalsy();
    });
});
```

Pilier #1 - Structure

Mentimeter

```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
}

describe("needToSleep", () => {
    it("should not need to sleep when it's day time even if tired", () => {
        // Arrange
        jest.spyOn(PlanningService.prototype, 'isNight', 'get').mockImplementation(() => false);
        baby.isTired = false;

        // Act
        const result = baby.needToSleep;

        // Assert
        expect(result).toBeFalsy();
    });
});
```

Pilier #1 - Structure - Act

Mentimeter

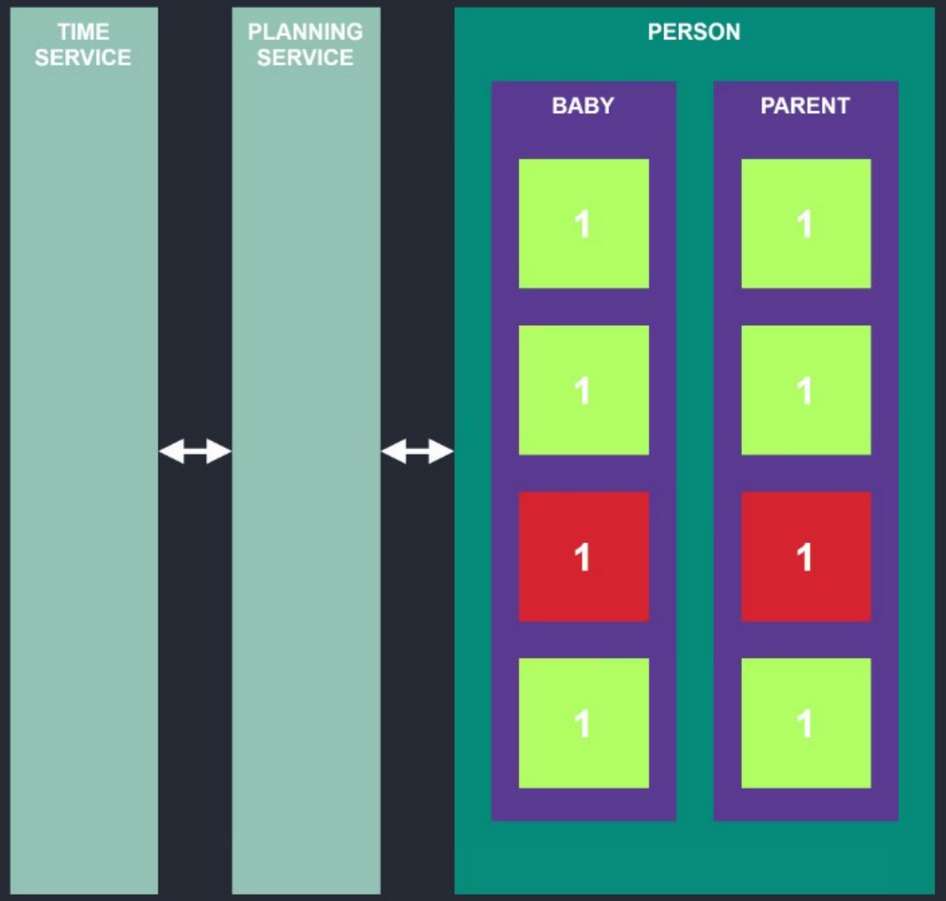
```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
}

describe("needToSleep", () => {
    it("should not need to sleep when it's day time even if tired", () => {
        // Arrange
        jest.spyOn(PlanningService.prototype, 'isNight', 'get').mockImplementation(() => false);
        baby.isTired = false;

        // Act && Assert
        expect(baby.needToSleep).toBeFalsy();
    });
});
```

Pilier #1 - Structure - Act

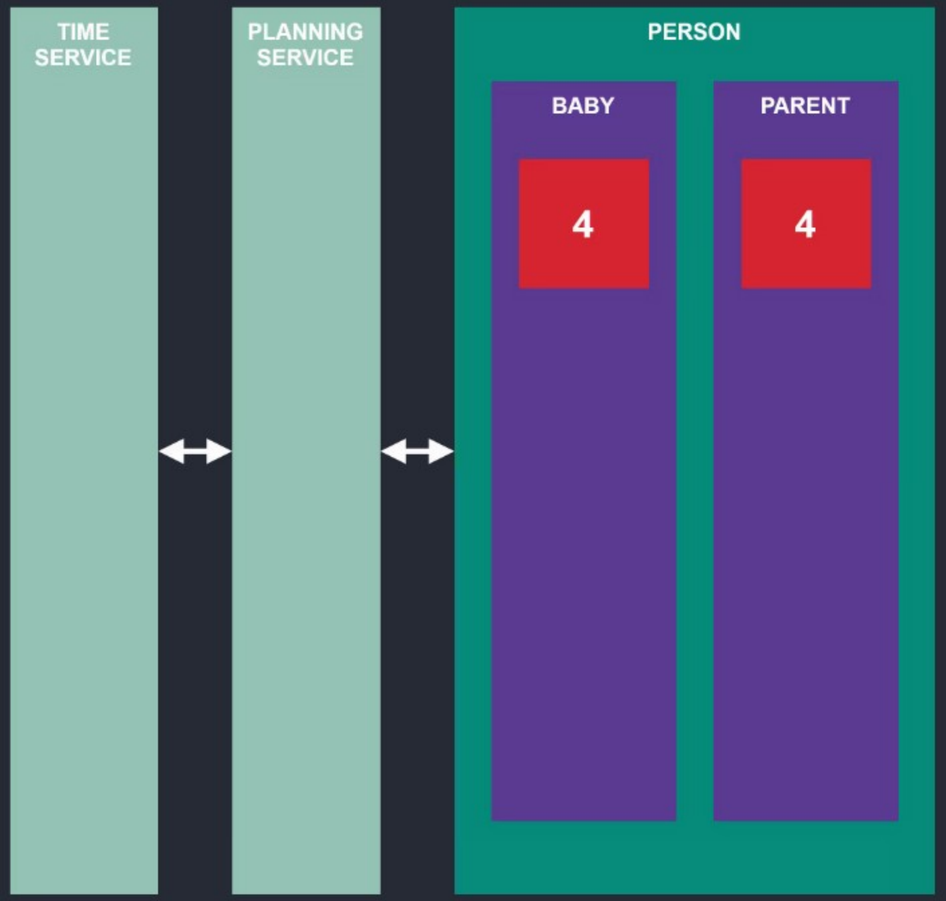
Mentimeter

```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
}

describe("needToSleep", () => {
    it("should not need to sleep when it's day time even if tired", () => {
        // Arrange
        jest.spyOn(PlanningService.prototype, 'isNight', 'get').mockImplementation(() => false);
        baby.isTired = false;

        // Act
        const result = baby.needToSleep;

        // Assert
        expect(result).toBeFalsy();
    });
});
```

Pilier #1 - Structure - Assert



Pilier #1 - Structure - Assert

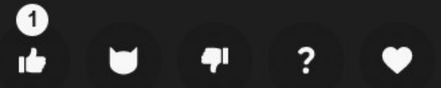


Pilier #1 - Structure - Assert

Mentimeter

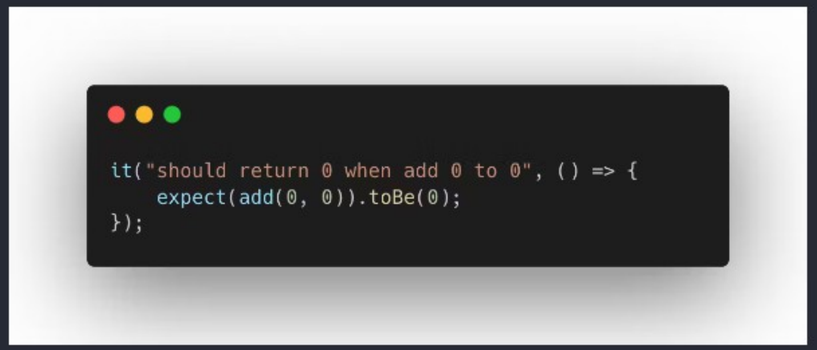
```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
}

describe("needToSleep", () => {
    it("should not need to sleep when it's day time even if tired", () => {
        // Arrange
        jest.spyOn(PlanningService.prototype, 'isNight', 'get').mockImplementation(() => false);
        baby.isTired = false;

        // Act
        const result = baby.needToSleep;

        // Assert
        expect(result).toBeFalsy();
    });
});
```

Pilier #1 - Structure - Autonomie



Mentimeter

La structure de ce test unitaire est correcte



```
it("should return 0 when add 0 to 0", () => {
    expect(add(0, 0)).toBe(0);
});
```

Vrai

8

Faux

9

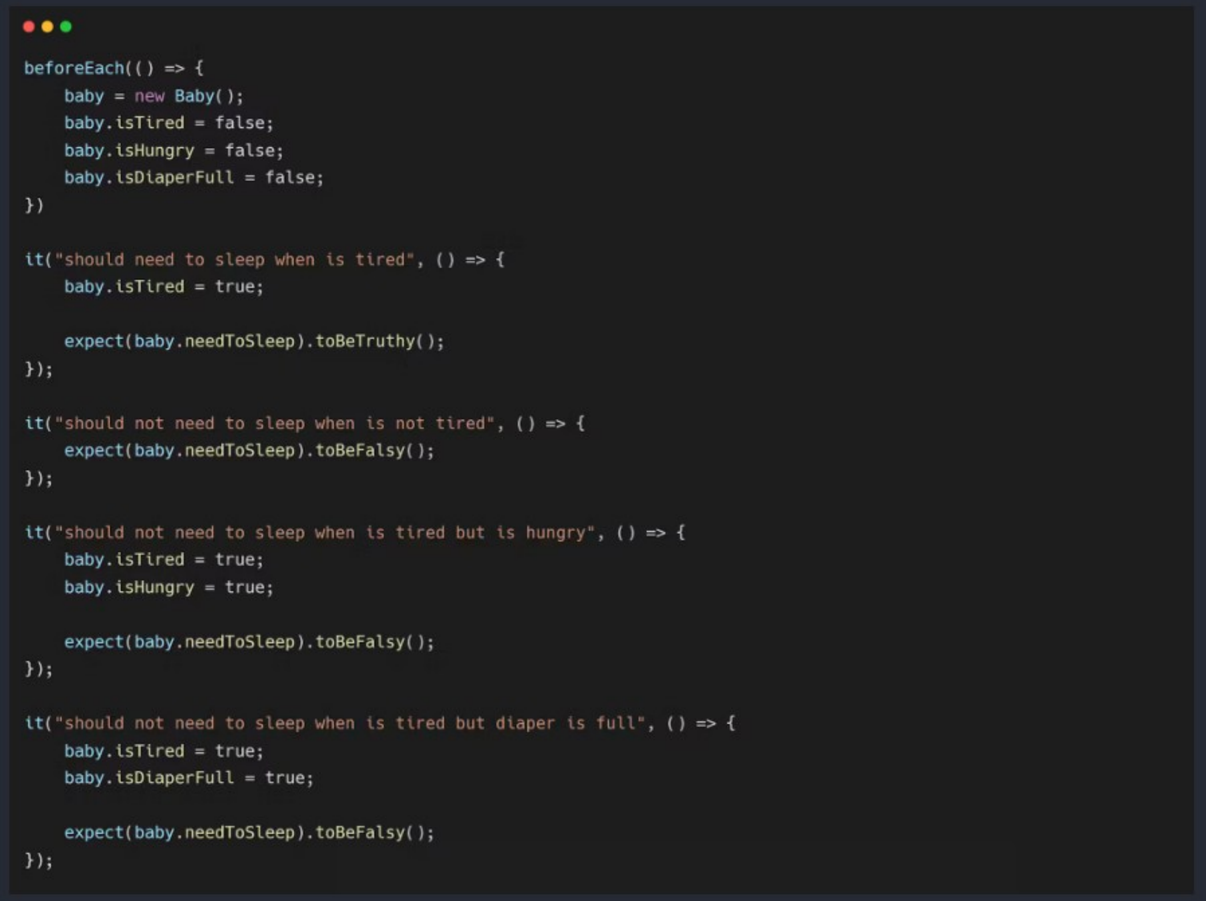
3 1 1 17

Mention

Mentimeter

```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
}

describe("needToSleep", () => {
    it("should not need to sleep when it's day time even if tired", () => {
        jest.spyOn(PlanningService.prototype, 'isNight', 'get').mockImplementation(() => false);
        baby.isTired = false;

        const result = baby.needToSleep;

        expect(result).toBeFalsy();
    });
});
```

Pilier #2 - Mention



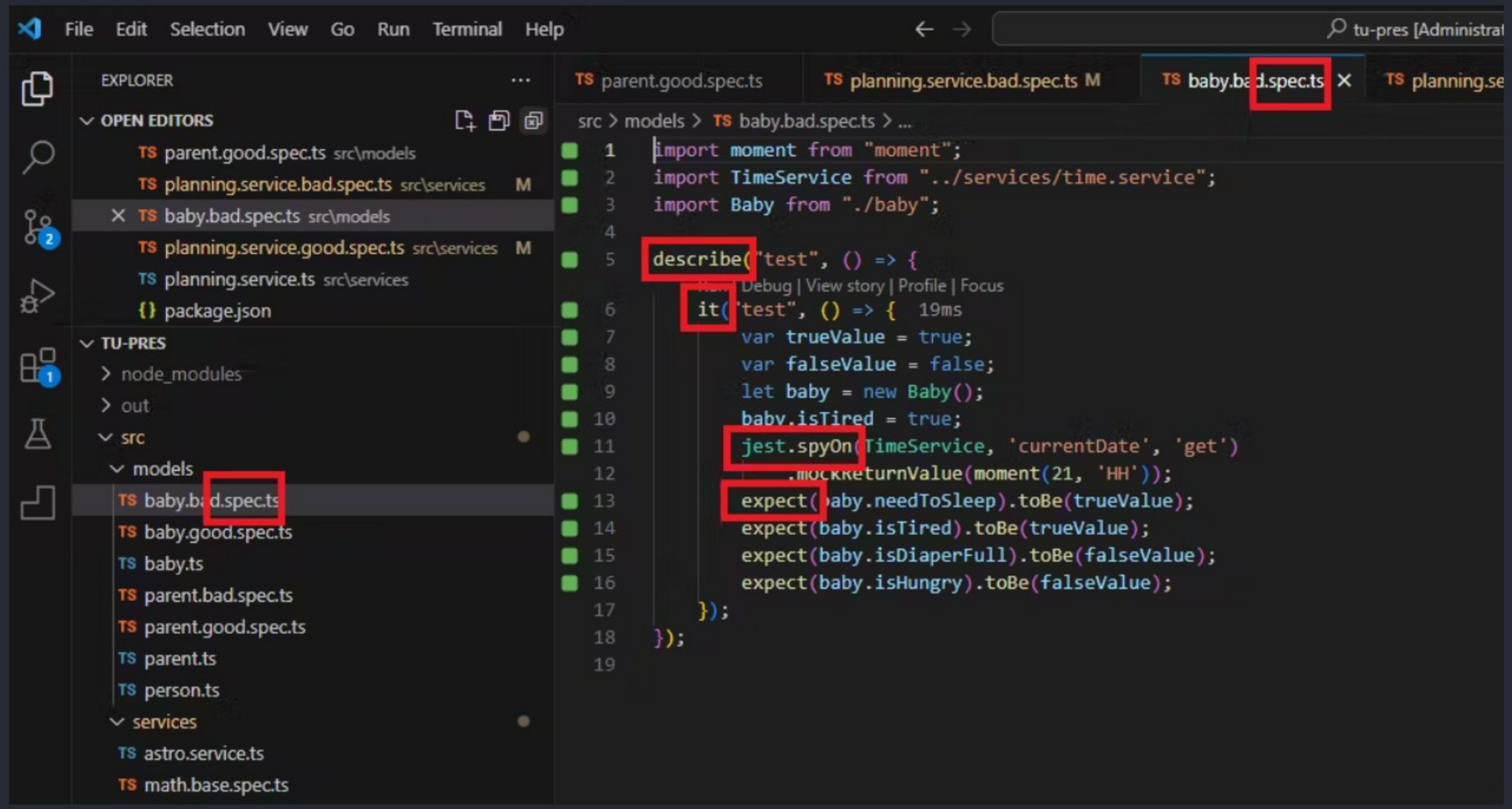
```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
    baby.isHungry = false;
    baby.isDiaperFull = false;
})

it("should need to sleep when is tired", () => {
    baby.isTired = true;

    expect(baby.needToSleep).toBeTruthy();
});

it("should not need to sleep when is not tired", () => {
    expect(baby.needToSleep).toBeFalsy();
});

it("should not need to sleep when is tired but is hungry", () => {
    baby.isTired = true;
    baby.isHungry = true;

    expect(baby.needToSleep).toBeFalsy();
});

it("should not need to sleep when is tired but diaper is full", () => {
    baby.isTired = true;
    baby.isDiaperFull = true;

    expect(baby.needToSleep).toBeFalsy();
});
```

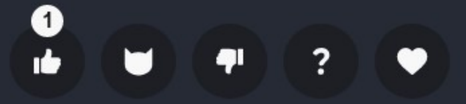
Pilier #2 - Mention

Mentimeter

```
beforeEach(() => {
    baby = new Baby();
    baby.isTired = false;
    baby.isHungry = false;
    baby.isDiaperFull = false;
})

it("should need to sleep when is tired", () => {
});

it("should not need to sleep when is not tired", () => {
});

it("should not need to sleep when is tired but is hungry", () => {
});

it("should not need to sleep when is tired but diaper is full", () => {
});
```

Pilier #2 - Mention


File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
parent.good.spec.ts src\models
planning.service.bad.spec.ts src\services M
baby.bad.spec.ts src\models
planning.service.good.spec.ts src\services M
planning.service.ts src\services
package.json
TU-PRES
node_modules
out
src
models
baby.bad.spec.ts
baby.good.spec.ts
baby.ts
parent.bad.spec.ts
parent.good.spec.ts
parent.ts
person.ts
services
astro.service.ts
math.base.spec.ts
parent.good.spec.ts
planning.service.bad.spec.ts M
baby.bad.spec.ts
src > models > baby.bad.spec.ts > ...
import moment from "moment";
import TimeService from "../services/time.service";
import Baby from "./baby";
describe("test", () => {
Debug | View story | Profile | Focus
it("test", () => { 19ms
var trueValue = true;
var falseValue = false;
let baby = new Baby();
baby.isTired = true;
jest.spyOn(TimeService, 'currentDate', 'get')
.mockReturnValue(moment(21, 'HH'));
expect(baby.needToSleep).toBe(trueValue);
expect(baby.isTired).toBe(trueValue);
expect(baby.isDiaperFull).toBe(falseValue);
expect(baby.isHungry).toBe(falseValue);
});
});


1

La mention de ce test unitaire est correcte

```
it("test that parent need to sleep when is tired", () => {
    const parent = new Parent();
    parent.isTired = true;

    jest.spyOn(TimeService, 'currentDate', 'get')
        .mockReturnValue(moment(21, 'HH'));

    expect(parent.needToSleep).toBeTruthy();
});
```

Vrai

0

Faux

11

11

Scope

```
describe("test", () => {
    it("test", () => {
        var trueValue = true;
        var falseValue = false;
        let baby = new Baby();
        baby.isTired = true;
        jest.spyOn(TimeService, 'currentDate', 'get')
            .mockReturnValue(moment(19, 'HH'));
        expect(baby.needToSleep).toBe(falseValue);
        expect(baby.isTired).toBe(trueValue);
    });
});
```

Pilier #3 - Scope

MOCK

Pilier #3 - Scope

Conclusion

- Structure
- Mention
- Scope


AA
chat.openai.com
Mentimeter
Confirmati..
Mon comp...
Connexion...
Acheter u...
Private Nu...
Accueil A...
ChatGPT 3.5
it('should return 0 when one of the numbers is 0', () => {
const result1 = sum(0, 5);
const result2 = sum(-3, 0);
expect(result1).toEqual(5);
});
Comment TU (Tests Unitaires) peux laisser un monde meilleur à tes enfants ?
Message ChatGPT...
Chat GPT
ChatGPT can make mistakes. Consider checking important information.