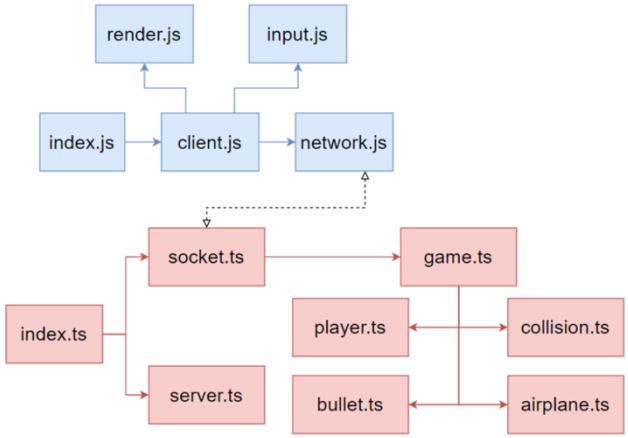
# Groupwork 3

- Group20
- Member : 0716217, 0716015, 0716047, 0616086

## Goal

- Develop a PVP .io game by TDD

## Architecture

- 


## Test case

### [1] Server

#### 1. index

##### (1) socket

```
describe('socket server', () => {
    it('can work', () => {
        # clientSocket accept serverSocket.emit correctly
    })
})
```

##### (2) server

```
describe('a express server', () => {
  it('should provide robots.txt', async)
  it('should provide index.html with a canvas whose id is canvas', async)
})
```

## 2. game

### (1) Object

- player

```
describe('Player class', () => {
  it('can get username and id')
  it('can accept update and send it to client with socketio', () => {
    # ################################################ #
    # Mock socket.emit as mockEmit                     #
    # case1:                                           #
    #    player.update(event)                          #
    #    expect(mockEmit).toBeCalledWith(GAME_UPDATE)#
    # case2:                                           #
    #    player.update(isDead)                         #
    #    expect(mockEmit).toBeCalledWith(GAME_OVER)  #
    # ################################################ #
  })
  it('player can be closed after disconnetion')
})
```

- airplane

```
describe('Airplane', () => {
    context('Init')
    context('update', () => {
        it('Health bar decreases upon accepting damage')
        it('Update position on a interval time, which should be in a map')
    })
    context('Direction', () => {
        it('Trun left')
        it('Go straight')
        it('Trun right')
    })
})
```

- bullet

```
describe('bullet', () => {
    it('construct a bullet')
    it('Set new position')
    it('Set new speed')
    it('Should update the location after a interval of time')
})
```

### (2) Game

- game

```
describe('Game', () => {
    context('Update the objects', () => {
        it('Should update the game on an interval')
        it('Each airplane needs update', () => {
            # spyOn airplanes
        })
        it('Each bullet needs update', () => {
            # spyOn bullets
        })
        it('Send update to each player'), () => {
            # spyOn(player, 'update').mockImplementation
        }
    })

    context('Remove the objects', () => {
        it('Add a player with player id')
        it('Remove the airplanes when they die, along with their players')
        it('Update players when they are removed')
        it('Bullet vanish when hit other player')
        it('Remove the players if they are disconnect to the game')
    })

    context('Player Input Handling', () => {
        it('Mouse click trigger a bullet release')
        it('Left arrow key press/release event set the left key status')
        it('Right arrow key press/release event set the right key status')
    })

})
```

- collision

```
describe('collision', () => {
    it('Should apply damage when bullet collides with player', () => {
        # spyOn airplane
        # checkCollisionToBullets([airplane], [bullet])
        # expect(airplane.acceptDamage).toHaveBeenCalledTimes(1)
    })
    it('Should not collide when outside radius')
    it('The bullet should not collide with it own player')
```

## [2] Client

### 1. index

```
describe('Index page logic', () => {
    it('Show loading, create client with username,
        and try to start after click enter btn')
    it('Hide the loading and show error msg when fails to start,
        and hide err msg after retry')
})
```

### 2. network

```
describe('network manager', () => {
    it('can create socket io connection')
    it('throw exception if connection fail')
})
```

### 3. input

```
describe('Game input manage module', () => {
    it('attach key&mouse event to send sock event')
    it('can dettach handlers')
})
```