trabalho.html estilo.css

```
Trabalho integrador{
    [BatalhaNaval.java]

        < Gustavo Martins, Ícaro Botelho e
        Rafael Pereira >

}
```

trabalho.html

estilo.css

Table Of ‘Contents’ {

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

01 {

[Ideia_Inicial]

< Criando a lógica e
estabelecendo o jogo >

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

```
esqueceTabuleiro(); {

    //Fazer um tabuleiro com matrizes seria algo mais
    difícil.

        <p Optamos por uma solução mais direta ao ponto
        para implementar os navios e as jogadas. >

        </p>

}
```

TrabalhoIntegrador.java

trabalho.html

estilo.css

Não existe tabuleiro!{

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

Criando o jogo ‘sem tabuleiro’ {

- **Step 01** A partir do momento que consideramos que o “tabuleiro” sempre terá água, qualquer coisa que não seja um navio será água.
- **Step 02** Dessa forma temos a base mais importante da lógica do jogo.
- **Step 03** Armazenar as coordenadas dos navios,
- **Step 04** Armazenar as coordenadas dos ataques,

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

Criando o jogo ‘sem tabuleiro’ {

- **Step 05** — Se a coordenada de um ataque bate com a coordenada de um navio → Acerto!
- **Step 06** — Cada jogador cria sua lista de jogadas e embarcações
- **Step 07** — O jogo envia a lista de jogadas de cada jogador
- **Step 08** — O jogo compara as coordenadas que recebeu com as embarcações que já tem

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

Criando o jogo ‘sem tabuleiro’ {

Step 09

Em momento nenhum os jogadores enviam as coordenadas de suas embarcações.

Step 10

O jogo retorna uma mensagem de acerto ou erro e atualiza o tabuleiro.

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

```
02 {

    [Classes]

        < Definindo o que cada classe
        faz e como interagem >

}
```

TrabalhoIntegrador.java

trabalho.html

estilo.css

Embarcacao {

< Posicionamento e ações dos barcos >

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

Agua {

< Tudo no tabuleiro que não seja barco >

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

```
Elemento {
    < Classe auxiliar geral >
}
```

TrabalhoIntegrador.java

trabalho.html

estilo.css

Coordenada {

< Classe auxiliar para Embarcacao >

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

Batalha {

< Main, chamada de todas as outras classes >

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

```
1
2
3
4
5  EnviarJogadaServidor{
6
7
8      < Fica responsável de client para client >
9
10 }
11
12
13
14
```

TrabalhoIntegrador.java

trabalho.html

estilo.css

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

ReceberJogadas {

< Fica responsável por receber as jogadas e adicionar na lista >

}

TrabalhoIntegrador.java

trabalho.html

estilo.css

```
03 {

    [Problemas_Encontrados]

        < Dificuldades encontradas que
        atrasaram e até impossibilitaram o
        desenvolvimento >

}
```

TrabalhoIntegrador.java

trabalho.html estilo.css

```
Problemas; {

  Socket é bidimensional?

  Como iniciar o Servidor e Batalha?

  Utilizar o mesmo client?


}
```

TrabalhoIntegrador.java

trabalho.html

estilo.css

```
Aprendizados; {

    [  ]  Conheça sua ferramenta.

        [  ]  Não foque no todo.

            [  ]  Use o que você sabe.

}
```

TrabalhoIntegrador.java

trabalho.html

estilo.css

< “Dívida técnica é uma metáfora criada pelo consultor americano Ward Cunningham para se referir a artefatos imaturos inseridos no software. Essa dívida acarreta juros que irão resultar em custos extras para a evolução e a manutenção do software, geralmente por meio da reescrita de código.” >

— Glauco

TrabalhoIntegrador.java

trabalho.html

estilo.css

Obrigado pela atenção!{

}

TrabalhoIntegrador.java