1. Processi primari

1.1 Fornitura

1.1.1 Studio di fattibilità

1.1.2 Rapporti di fornitura con la Proponente Red Babel

1.1.3 Documentazione fornita

1.2 Sviluppo

1.2.1 Analisi dei Requisiti

1.2.2 Codifica

In questa sotto-sezione vengono elencate le norme alle quali i Programmatori devono attenersi durante l'attività di programmazione e implementazione.
Ogni norma è rappresentata da un paragrafo. Ciascuna ha un titolo, una breve descrizione e, se necessario, un esempio che illustra i modi accettati o meno. Alcune di esse includono inoltre una lista di possibili eccezioni d'uso. L'uso di norme e convenzioni è fondamentale per permettere la generazione di codice leggibile e uniforme, agevolare le fasi di manutenzione, verifica e validazione e migliorare la qualità del prodotto.

1.2.2.1 Javascript

1.2.2.2 React

1.2.2.3 Solidity

In questa sotto-sezione vengono elencate le norme tratte dalla **documentazione officiale di Solidity**[1].

Identazione: vanno utilizzati quattro (4) spazi per ogni livello di indentazione.

Spazi: è vietato l'utilizzo di tabulazioni, che devono essere necessariamente sostituite da spazi. Al fine di assicurare il rispetto di questa regola si consiglia di configurare adeguatamente il proprio editor o IDE.

Linee vuote 1: devono essere lasciate due (2) linee vuote tra la dichiarazione di più smart contract.

SI

```
contract A {
...
}


contract B {
...
}


contract C {
...
}
```

NO

```
contract A {
...
}
contract B {
...
}

contract C {
...
}
```

Linee vuote 2: le linee vuote possono essere omesse tra le dichiarazioni di funzioni in linea.

[1] http://solidity.readthedocs.io/en/develop/style-guide.html

SI

```
contract A {
        function spam();
        function ham();
}


contract B is A {
        function spam() {
        ...
        }

        function ham() {
        ...
        }
}
```

NO

```
contract A {
        function spam() {
        ...
        }
        function ham() {
        ...
        }
}
```

Codifica dei codici sorgenti: i codici sorgente devono essere codificati in UTF-8.

Imports: tutte le dichiarazioni di import vanno fatte all'inizio del file.

SI

```
import "owned";


contract A {
...
}


contract B is owned {
...
}
```

NO

```
contract A {
...
}


import "owned";


contract B is owned {
...
}
```

Ordine delle funzioni: All'interno di un contratto l'ordine è fondamentale per identificare meglio quali funzioni è possibile chiamare e per trovare più velocemente le definizioni del costruttore e di eventuali funzioni di fallback.

Le funzioni devono essere ragruppate secondo la loro visibilità e ordinate in questo modo:

- costruttore
- funzioni di fallback (se esistono)
- esterne
- pubbliche
- interne
- private

In ogni gruppo le funzioni costanti vanno dichiarate per ultime.

SI

```
contract A {
        function A() {
        ...
        }

        function() {
        ...
        }

        // External functions
        // ...

        // External functions
        // that are constant
        // ...

        // Public functions
        // ...

        // Internal functions
        // ...

        // Private functions
        // ...
}
```

NO

```
contract A {

        // External functions
        // ...

        // Private functions
        // ...

        // Public functions
        // ...

        function A() {
        ...
        }

        function() {
        ...
        }

        // Internal functions
        // ...
}
```

Spazi bianco nelle espressioni: non vanno messi spazi bianchi nelle seguenti situazioni:

- immediatamente dopo l'apertura o la chiusura di una parentesi tonda, quadrata o graffa con un'espressione di una singola riga

SI

```
spam(ham[1], Coin({name: "ham"}));
```

NO

```
spam( ham[ 1 ], Coin( { name: "ham" } ) );
```

1.2.2.4 Scss

1.2.3 Progettazione