



Allegato tecnico

Sons Of SWE - Progetto Marvin

sonsofswe.swe@gmail.com

Informazioni sul documento

| | |
|---------------|---|
| Uso | Esterno |
| Distribuzione | Vardanega Tullio Cardin Riccardo Gruppo Sons Of SWE |

Descrizione

Documento che ha lo scopo di illustrare la Product Baseline, ponendo attenzione alle scelte architettrurali e alla copertura di use case e requisiti funzionali.



Indice

| | | |
|----------|--|-----------|
| 1 | Introduzione | 4 |
| 1.1 | Scopo del documento | 4 |
| 1.2 | Scopo del prodotto | 4 |
| 1.3 | Glossario | 4 |
| 1.4 | Riferimenti | 4 |
| 1.4.1 | Normativi | 4 |
| 1.4.2 | Informativi | 4 |
| 2 | Requisiti di sistema | 5 |
| 3 | Installazione ed esecuzione | 6 |
| 4 | Confronto con il Poc | 8 |
| 5 | Architettura del prodotto | 9 |
| 5.1 | Model | 9 |
| 5.1.1 | Design patterns | 9 |
| 5.1.2 | Diagramma delle classi | 9 |
| 5.2 | View | 10 |
| 5.2.1 | Design patterns | 10 |
| 5.2.2 | Diagramma delle classi | 10 |
| 5.3 | ViewModel | 10 |
| 5.3.1 | Design patterns | 10 |
| 5.3.2 | Diagramma delle classi | 10 |
| 5.4 | Diagrammi di sequenza | 11 |
| 5.4.1 | Login | 11 |
| 5.4.2 | Inserimento di un utente | 11 |
| 5.4.3 | Inserimento di un anno accademico | 11 |
| 5.4.4 | Visualizzazione di un anno accademico | 11 |
| 6 | Requisiti soddisfatti | 12 |
| 6.1 | Tabella del soddisfacimento dei requisiti | 12 |
| 6.2 | Grafici sui requisiti soddisfatti | 13 |
| 7 | Casi d'uso soddisfatti | 14 |
| 7.1 | Tabella del soddisfacimento dei casi d'uso | 14 |
| 7.2 | Grafici sui casi d'uso soddisfatti | 15 |



Elenco delle figure

| | | |
|----|---|----|
| 1 | Setta l'RPC inserendo http://localhost:9545 | 6 |
| 2 | Clicca su Import Existing DEN | 7 |
| 3 | Inserisci la seed phrase e la password che vuoi usare | 7 |
| 4 | Diagramma generale dei package | 9 |
| 5 | Diagramma delle classi del componente Model | 10 |
| 6 | Diagramma delle classi del componente View | 10 |
| 7 | Diagramma delle classi del componente ViewModel | 10 |
| 8 | Diagramma di sequenza del processo di login | 11 |
| 9 | Diagramma di sequenza del processo di inserimento di un utente | 11 |
| 10 | Diagramma di sequenza del processo di inserimento di un anno accademico | 11 |
| 11 | Diagramma di sequenza del processo di visualizzazione di un anno accademico | 12 |
| 12 | Requisiti soddisfatti | 13 |
| 13 | Requisiti obbligatori soddisfatti | 13 |
| 14 | Casi d'uso soddisfatti | 15 |



Elenco delle tabelle

| | | |
|---|---|----|
| 1 | Confronto tra Proof of Concept e Product Baseline | 8 |
| 2 | Requisiti soddisfatti | 12 |
| 3 | Casi d'uso soddisfatti | 14 |



1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di descrivere gli obiettivi di qualità, di processo e di prodotto da raggiungere nella realizzazione del progetto e le strategie di verifica e validazione adottate per il raggiungimento di tali obiettivi.

1.2 Scopo del prodotto

Lo scopo del prodotto è quello di realizzare un *prototipo_G* di Uniweb come una *DApp_G* in esecuzione su *Ethereum_G*. I cinque attori principali che si rapportano con Marvin sono:

- Utente non autenticato;
- Università;
- Amministratore;
- Professore;
- Studente.

Il portale deve quindi permettere agli studenti di accedere alle informazioni riguardanti le loro carriere universitarie, di iscriversi agli esami, di accettare o rifiutare voti e di poter vedere il loro libretto universitario. Ai professori deve invece essere permesso di registrare i voti degli studenti. L'università ogni anno crea una serie di corsi di laurea rivolti a studenti, dove ognuno di essi comprende un elenco di esami disponibili per anno accademico. Ogni esame ha un argomento, un numero di crediti e un professore associato. Gli studenti si iscrivono ad un corso di laurea e tramite il libretto elettronico mantengono traccia ufficiale del progresso.

1.3 Glossario

Nel documento *Glossario.v1.0.0* i termini tecnici, gli acronimi e le abbreviazioni sono definiti in modo chiaro e conciso, in modo tale da evitare ambiguità e massimizzare la comprensione dei documenti.

I vocaboli presenti in esso saranno posti in corsivo e presenteranno una "G" maiuscola a pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Capitolato d'appalto C6 - Marvin: dimostratore di Uniweb su Ethereum** <http://www.math.unipd.it/~tullio/IS-1/2017/Progetto/C6.pdf>;
- *NormeDiProgetto_v4.0.0*;

1.4.2 Informativi

- *AnalisiDeiRequisiti_v4.0.0*;



2 Requisiti di sistema

Per l'installazione e l'utilizzo di questo software sono richiesti alcuni prerequisiti:

- Browser web Google Chrome (aggiornato alla versione 60 o superiori) o Mozilla Firefox (aggiornato alla versione 50 o superiori);
- Plugin Metamask (aggiornato alla versione 4.7.1 o superiori) per i browser di cui sopra;
<https://metamask.io/>
- Git
<https://git-scm.com/downloads>
- Python aggiornato alla versione 2.7;
<https://www.python.org/downloads/>
- Node package manager alla versione 6, e Node alla 8.11.2
<https://nodejs.org/it/>
- Nel caso si utilizzi Windows sarà necessario installare *windows-build-tools* digitando nella powershell:

```
1  npm install --global --production windows-build-tools
```



3 Installazione ed esecuzione

Il codice relativo alla Product Baseline lo si può trovare al seguente link:

[linkAllaRepo](#)

Una volta fatto il clone della repository o dopo aver scaricato lo zip, sono necessari i seguenti passi per far partire l'applicazione:

1. Posizionarsi nella root della repo ed eseguire nella shell:

```
1 npm install -g ganache-cli
2 npm install -g truffle
3 npm i
```

2. In seguito sempre nella shell:

```
1 ./startBlockchain.ps1
```

3. Infine è necessario eseguire:

```
1 ./loadProject.ps1
```

A questo punto noterai che il tuo browser predefinito ha aperto automaticamente l'homepage. Ora dovrai connetterti a Metamask: nel tuo browser clicca sull'icona di Metamask e accetta l'informativa sulla privacy e le condizioni d'uso. Poi clicca su **Main Network** e scegli **Custom RPC**, inserisci nel primo form **http://localhost:9545** come in Figura 1 e clicca su **Save**.

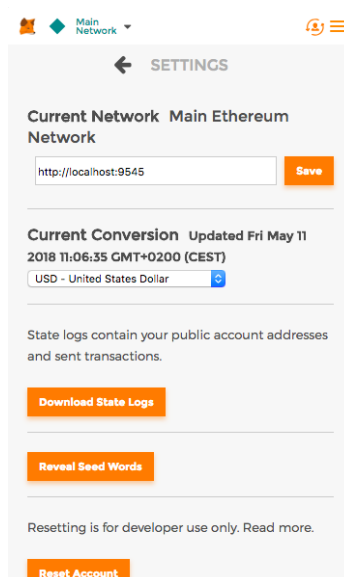


Figura 1: Setta l'RPC inserendo **http://localhost:9545**

Ora, come in Figura 2, clicca su **Import Existing DEN** e (vedi Figura 3) inserisci la frase **candy maple cake sugar pudding cream honey rich smooth crumble sweet treat** e la password che vuoi usare per il tuo account.



The image shows the Metamask 'Encrypt your new DEN' screen. At the top, it says 'METAMASK' and 'Encrypt your new DEN' with a question mark icon. Below this are two input fields: 'New Password (min 8 chars)' and 'Confirm Password'. At the bottom, there is an orange 'CREATE' button and a link that says 'Import Existing DEN'.

Figura 2: Clicca su **Import Existing DEN**

The image shows the Metamask 'RESTORE VAULT' screen. At the top, it says 'RESTORE VAULT' and 'Main Network'. Below this is a 'Wallet Seed' section with a text box containing the seed phrase: 'candy maple cake sugar pudding cream honey rich smooth crumble sweet treat'. Below the text box are two input fields for a password, each with a series of dots. At the bottom, there are two orange buttons: 'CANCEL' and 'OK'.

Figura 3: Inserisci la seed phrase e la password che vuoi usare

4 Confronto con la Poc

In precedenza è stato realizzata una *Proof of Concept* in cui si è data dimostrazione delle tecnologie che si sarebbero dovute utilizzare per la realizzazione del progetto Marvin. Tale produzione è disponibile all'indirizzo: <https://github.com/SOS-SonsOfSwe/Marvin-PoC>. Durante la ricerca di strumenti adatti allo scopo il team ha deciso inizialmente di basare la prima produzione del prodotto sul framework *Truffle*, dal momento che esso presentava molte delle tecnologie di interesse. Con uno studio più approfondito del pacchetto il gruppo si è reso conto che esso era una buona base di partenza per la realizzazione di un prodotto più complesso, sebbene necessitasse di alcune correzioni strutturali. Qui di seguito si elencano i limiti riscontrati e le soluzioni adottate:

| Proof of Concept | Product Baseline |
|---|---|
| Architettura ben strutturata ma male incapsulata | Organizzazione migliore della struttura del pacchetto a fronte di una conoscenza più approfondita del design pattern MVVM e di altri su cui il framework si basa |
| Interfaccia povera e strutturata in maniera confusionaria, non chiara e difficilmente intuibile | Ristrutturazione completa e incremento dell'interfaccia utente, ora provvista di tutte le parti atte alla soddisfazione della maggior parte dei requisiti opzionali e obbligatori |
| Database solidity estremamente scarno e con poche funzionalità di sicurezza e di ottimizzazione | Strutturazione avanzata del backend di solidity, decisamente più ottimizzato e performante; implementazione già in fase di completamento |

Tabella 1: Confronto tra Proof of Concept e Product Baseline

5 Architettura del prodotto

Dopo un approfondito studio il team ha optato per l'utilizzo del design pattern **Model View View-Model**, che prevede tre macrosezioni:

- **Model:** rappresenta i dati contenuti nel sito, ma non i comportamenti o i servizi che manipolano l'informazione. Non è responsabile della renderizzazione;
- **View:** si occupa di rappresentare le informazioni contenute nel sito ed è quello con cui l'utente interagisce; la view in questo design pattern è attiva, al contrario di quello che succede nell'MVC, questo perché contiene comportamenti, eventi e riferimenti a stati del sito, che quindi presuppongono una conoscenza della logica che sta dietro ai dati;
- **Viewmodel:** fornisce i dati dal Model in una forma in cui la View può usufruirne. Si occupa inoltre della logica della vista e di mantenersi costantemente sincronizzato con la View.

Per poter apprezzare questa suddivisione nel pacchetto del team viene riportato qui di seguito il diagramma generale dei package. Si proseguirà successivamente alla descrizione delle implementazioni di ogni sezione in relazione all'architettura della Product Baseline, illustrandone i rispettivi design pattern utilizzati.

Per facilitare la comprensione della struttura dei diagrammi delle classi, il team ha deciso di raggruppare tutti i *Containers*, i *Components* e le *actions* nei loro rispettivi packages. Infatti il comportamento di ogni classe di ogni package è modulare:

- Tutti i *Components* importano la classe *React.Component*
- Tutti i *Containers* importano il rispettivo *Component*, la rispettiva *Action* e il pacchetto *react-redux*.
- Tutte le *actions* importano le *costants* che azionano il *reducer*, il pacchetto *react-router*, lo *store*, il pacchetto *truffle-contract* e la classe *IpfsUtils*

I diagrammi presentati in questo documento illustrano la struttura sinteticamente; la loro versione integrale si può trovare nella cartella **INSERIRE LA CARTELLA**.

INSERIRE DIAGRAMMA DEI PACKAGE IN GENERALE

Figura 4: Diagramma generale dei package

5.1 Model

5.1.1 Design patterns

Per quanto riguarda la macrosezione relativa alla componente Model, si sono utilizzati i seguenti design pattern: **INSERIRE I PATTERN**

- : ;
- : .
- : .

5.1.2 Diagramma delle classi

In seguito è riportato il diagramma delle classi relativo a questa sezione con evidenziato la posizione del design pattern utilizzato.

INSERIRE DIAGRAMMA CON VISTA SU MODEL

Figura 5: Diagramma delle classi del componente Model

5.2 View

5.2.1 Design patterns

Progettando questa sezione ci si è resi conto che si sarebbero potute generare delle classi *dumb*, ovvero slegate dalla logica del sistema e che si sarebbero dovute occupare solamente della rappresentazione delle informazioni. Per ottenere questo si è ricorso all'utilizzo del seguente design pattern:

- **Decorator**: sfruttando il macro-package *Container*, che poi si occupa di collegare il componente allo stato di *redux*, si possono decorare tutte le componenti ivi presenti con dei componenti react puri, ai quali vengono passate eventualmente le informazioni e le funzioni a disposizione attraverso un accesso al *this.props*. In questa maniera si ha la completa separazione tra rappresentazione e dati - cosa auspicata dal framework MVVM - e si facilitano modifiche future.

5.2.2 Diagramma delle classi

In seguito è riportato il diagramma delle classi relativo a questa sezione con evidenziato la posizione del design pattern utilizzato.

INSERIRE DIAGRAMMA CON VISTA SU VIEW

Figura 6: Diagramma delle classi del componente View

5.3 ViewModel

5.3.1 Design patterns

Per la sezione riguardante il ViewModel, si è ricorso all'utilizzo dei seguenti design pattern:

- **Observer**: descrizione per quanto riguarda il comportamento di trigger-client che opera Redux sullo stato di React;
- **Facade**: per generare un'interfaccia astratta dalle implementazioni delle classi che poi si vanno ad utilizzare;
- **Adapter**: per interagire col model si usa un adaptator offerto dal framework truffle che si occupa di fare-un-sacco-di-cose-belle-da-chiedere-a-Stefano;
- **Command**: come redux ordina a react di re-renderizzare le pagine al cambiamento dello state dovuto ad un dispatch di un'azione.

5.3.2 Diagramma delle classi

In seguito è riportato il diagramma delle classi relativo a questa sezione con evidenziato la posizione del design pattern utilizzato.

INSERIRE DIAGRAMMA CON VISTA SU VIEWMODEL

Figura 7: Diagramma delle classi del componente ViewModel

5.4 Diagrammi di sequenza

Come per i diagrammi delle classi, sono stati redatti i seguenti diagrammi di sequenza, disponibili e meglio visualizzabili nella cartella [INSERIRE LA CARTELLA](#);

5.4.1 Login

Il diagramma di sequenza riportato qui di seguito raffigura il processo di login, durante il quale l'utente che vuole accedere può essere autenticato dal sistema. [INSERIRE DIAGRAMMA DI LOGIN](#)

Figura 8: Diagramma di sequenza del processo di login

5.4.2 Inserimento di un utente

Il seguente diagramma di sequenza rappresenta l'azione di inserimento di un utente nel sistema. [INSERIRE DIAGRAMMA DI INSERIMENTO UTENTE](#)

Figura 9: Diagramma di sequenza del processo di inserimento di un utente

5.4.3 Inserimento di un anno accademico

Il diagramma di sequenza riportato qui di seguito raffigura il processo di inserimento di un nuovo anno accademico nel sistema. [INSERIRE DIAGRAMMA DI INSERIMENTO ANNO ACCADEMICO](#)

Figura 10: Diagramma di sequenza del processo di inserimento di un anno accademico

5.4.4 Visualizzazione di un anno accademico

Il seguente diagramma di sequenza rappresenta l'azione di visualizzazione di un anno accademico. [INSERIRE DIAGRAMMA DI VISUALIZZAZIONE ANNO ACCADEMICO](#)

Figura 11: Diagramma di sequenza del processo di visualizzazione di un anno accademico

6 Requisiti soddisfatti

6.1 Tabella del soddisfacimento dei requisiti

| ID requisito | Soddisfacimento nell'architettura | Soddisfacimento nel codice |
|--------------|-----------------------------------|----------------------------|
| R0F1 | SODDISFATTO | SODDISFATTO |
| R0F2 | SODDISFATTO | SODDISFATTO |
| R0F3 | SODDISFATTO | SODDISFATTO |
| R0F4 | SODDISFATTO | SODDISFATTO |
| R0F5 | SODDISFATTO | SODDISFATTO |
| R0F6 | SODDISFATTO | SODDISFATTO |
| R2F7 | SODDISFATTO | NON SODDISFATTO |
| R2F8 | SODDISFATTO | NON SODDISFATTO |
| R2F9 | SODDISFATTO | NON SODDISFATTO |
| R2F10 | SODDISFATTO | NON SODDISFATTO |
| R2F11 | SODDISFATTO | NON SODDISFATTO |
| R2F13 | SODDISFATTO | NON SODDISFATTO |
| R2F14 | SODDISFATTO | NON SODDISFATTO |
| R0F15 | SODDISFATTO | SODDISFATTO |
| R0F16 | SODDISFATTO | SODDISFATTO |
| R0F17 | SODDISFATTO | SODDISFATTO |
| R0F18 | SODDISFATTO | SODDISFATTO |
| R0F19 | SODDISFATTO | NON SODDISFATTO |
| R2F20 | SODDISFATTO | NON SODDISFATTO |
| R2F21 | SODDISFATTO | NON SODDISFATTO |
| R2F22 | SODDISFATTO | NON SODDISFATTO |
| R0F23 | SODDISFATTO | SODDISFATTO |
| R0F24 | SODDISFATTO | SODDISFATTO |
| R2F25 | SODDISFATTO | NON SODDISFATTO |
| R0F26 | SODDISFATTO | SODDISFATTO |

Tabella 2: Requisiti soddisfatti

6.2 Grafici sui requisiti soddisfatti

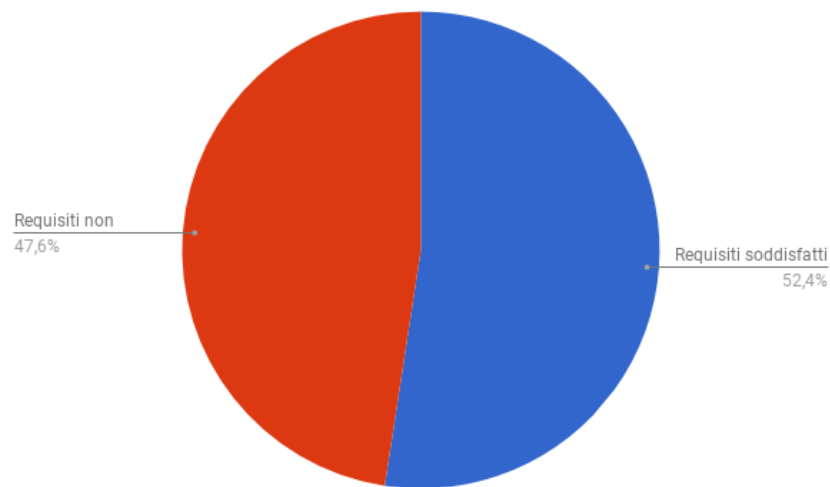


Figura 12: Requisiti soddisfatti

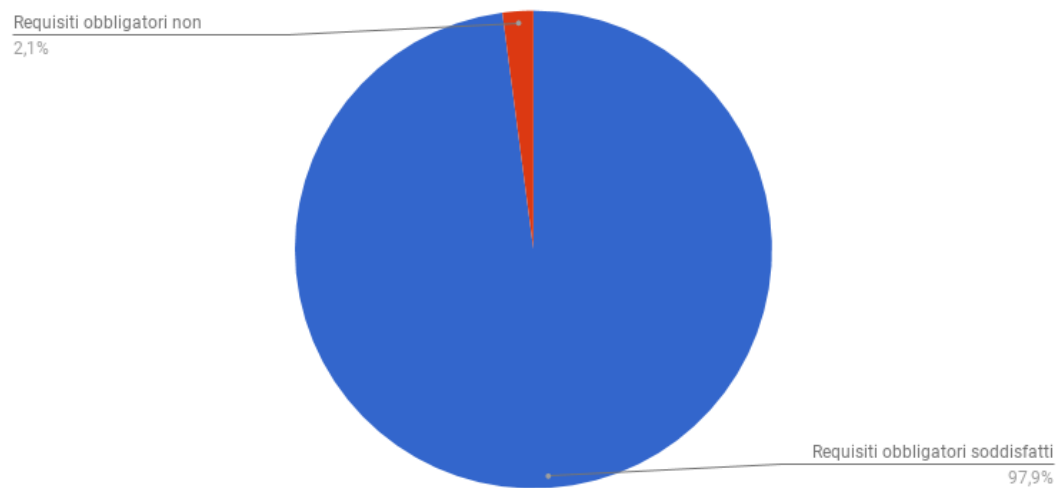


Figura 13: Requisiti obbligatori soddisfatti

7 Casi d'uso soddisfatti

7.1 Tabella del soddisfacimento dei casi d'uso

| ID caso d'uso | Soddisfacimento nell'architettura | Soddisfacimento nel codice |
|---------------|-----------------------------------|----------------------------|
| UC1 | SODDISFATTO | NON SODDISFATTO |
| UC2 | SODDISFATTO | NON SODDISFATTO |
| UC3 | SODDISFATTO | NON SODDISFATTO |
| UC4 | SODDISFATTO | NON SODDISFATTO |
| UC6 | SODDISFATTO | SODDISFATTO |
| UC7 | SODDISFATTO | SODDISFATTO |
| UC8 | SODDISFATTO | SODDISFATTO |
| UC9 | SODDISFATTO | SODDISFATTO |
| UC10 | SODDISFATTO | SODDISFATTO |
| UC11 | SODDISFATTO | NON SODDISFATTO |
| UC12 | SODDISFATTO | NON SODDISFATTO |
| UC13 | SODDISFATTO | NON SODDISFATTO |
| UC14 | SODDISFATTO | NON SODDISFATTO |
| UC15 | SODDISFATTO | NON SODDISFATTO |
| UC16 | SODDISFATTO | NON SODDISFATTO |
| UC17 | SODDISFATTO | NON SODDISFATTO |
| UC18 | SODDISFATTO | NON SODDISFATTO |
| UC19 | SODDISFATTO | NON SODDISFATTO |
| UC20 | SODDISFATTO | NON SODDISFATTO |
| UC21 | SODDISFATTO | NON SODDISFATTO |
| UC22 | SODDISFATTO | NON SODDISFATTO |
| UC23 | SODDISFATTO | SODDISFATTO |
| UC24 | SODDISFATTO | NON SODDISFATTO |
| UC25 | SODDISFATTO | NON SODDISFATTO |
| UC26 | SODDISFATTO | NON SODDISFATTO |
| UC27 | SODDISFATTO | SODDISFATTO |
| UC28 | SODDISFATTO | SODDISFATTO |
| UC29 | SODDISFATTO | NON SODDISFATTO |
| UC30 | SODDISFATTO | SODDISFATTO |
| UC31 | SODDISFATTO | SODDISFATTO |
| UC32 | SODDISFATTO | SODDISFATTO |
| UC33 | SODDISFATTO | SODDISFATTO |

Tabella 3: Casi d'uso soddisfatti

7.2 Grafici sui casi d'uso soddisfatti

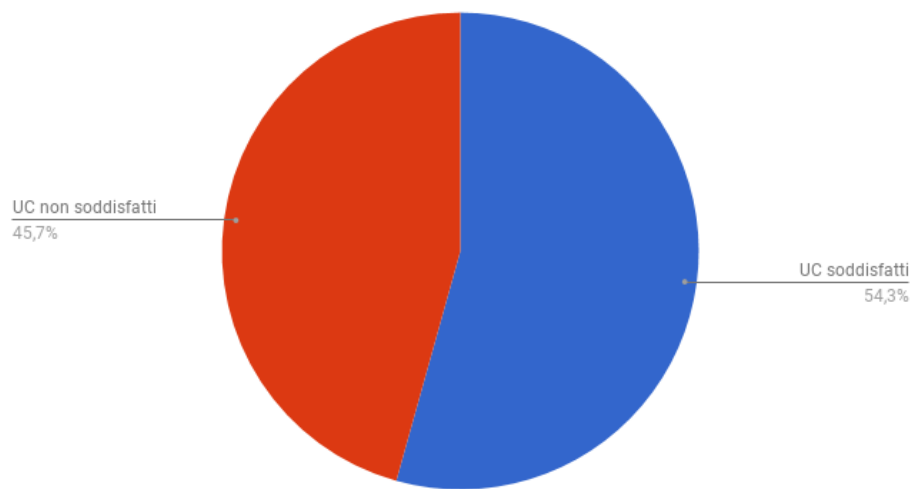


Figura 14: Casi d'uso soddisfatti