

# Movinator

Andrea Colato, Tommaso Bonetti

17 luglio 2016

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Modello</b>	<b>4</b>
<b>3</b>	<b>Specifica</b>	<b>5</b>
3.1	Requisiti . . . . .	5
<b>4</b>	<b>Scenari d'uso</b>	<b>6</b>
4.1	Use Case Diagram . . . . .	6
4.1.1	Attore Guest . . . . .	7
4.1.2	Attore User . . . . .	8
4.1.3	Attore Admin . . . . .	9
4.2	Sequence Diagram degli use case . . . . .	10
<b>5</b>	<b>Progettazione</b>	<b>12</b>
5.1	Progettazione Base di Dati . . . . .	12
5.1.1	Schema Concettuale . . . . .	12
5.1.2	Schema Logico . . . . .	12
5.1.3	Schema Fisico . . . . .	12
5.2	Class Diagram . . . . .	14
5.2.1	Site . . . . .	14
5.2.2	Malware . . . . .	15
5.2.3	User . . . . .	15
5.3	Sequence Diagram . . . . .	17
5.4	Activity Diagram . . . . .	18
<b>6</b>	<b>Architettura</b>	<b>19</b>
6.1	Lamp . . . . .	19
6.2	Yii 1.7 - PHP Framework . . . . .	19
6.3	Repository GitHub . . . . .	21
6.4	Qualità Architeturali . . . . .	21
<b>7</b>	<b>Sviluppo</b>	<b>22</b>
7.1	Realizzazione prototipo . . . . .	22
7.2	Realizzazione Saas . . . . .	22
<b>8</b>	<b>Test sul prototipo</b>	<b>23</b>
8.1	Push . . . . .	23
8.1.1	Push intero . . . . .	23
8.1.2	Push registro . . . . .	23
8.1.3	Push memoria . . . . .	24
8.2	Pop . . . . .	25
8.2.1	Pop variabile . . . . .	25
8.2.2	Pop registro . . . . .	25
8.2.3	Pop memoria . . . . .	25

8.3	Inc . . . . .	26
8.3.1	Inc registro . . . . .	26
8.3.2	Inc memoria . . . . .	27
8.4	Dec . . . . .	28
8.4.1	Dec registro . . . . .	28
8.4.2	Dec memoria . . . . .	29
8.5	Add . . . . .	30
8.5.1	Add intero, registro . . . . .	30
8.5.2	Add intero , memoria . . . . .	31
8.5.3	Add registro, registro . . . . .	32
8.5.4	Add memoria, registro . . . . .	33
8.5.5	Add registro, memoria . . . . .	34
8.6	Sub . . . . .	35
8.6.1	Sub intero, registro . . . . .	35
8.6.2	Sub intero, memoria . . . . .	36
8.6.3	Sub registro, registro . . . . .	37
8.6.4	Sub memoria, registro . . . . .	38
8.6.5	Sub registro, memoria . . . . .	39
<b>9</b>	<b>Test sull'applicazione</b>	<b>40</b>
9.1	Test funzionalità applicazione . . . . .	41
9.1.1	Controllo login . . . . .	41
9.1.2	Controllo registrazione . . . . .	42
9.1.3	Fallimento caricamento malware . . . . .	45
9.2	Test sicurezza applicazione . . . . .	46
<b>10</b>	<b>Design pattern</b>	<b>48</b>
10.1	MVC . . . . .	48
10.2	Singleton . . . . .	49
<b>11</b>	<b>Gestione evoluzione</b>	<b>50</b>

# Capitolo 1

## Introduzione

Movinator non è altro che un principio di concretizzazione di ciò che Stephen Dolan racconta nel paper "mov is Turing-complete". In questo paper viene dimostrato come un programma composto da sole *mov* e con un'unica *jmp* alla fine (per garantire la non terminazione) sia sufficiente per garantire la Turing-completezza.

Con la benedizione del Dott. Roberto Giacobazzi abbiamo deciso di creare uno strumento che abbia la capacità trasformare un'istruzione assembly in una sequenza di sole istruzioni *mov* facente le medesime operazioni.

Movinator è un semplice compilatore scritto inizialmente in Java con la successiva creazione di un SaaS per il progetto di Ingegneria del Software, che trasforma le seguenti istruzioni assembly in sequenze di *mov*:

- Push
- Pop
- Inc
- Dec
- Add
- Sub

Queste istruzioni assembly sono tradotte solo nella loro versione a 32 bit.

## Capitolo 2

# Modello

Un modello di sviluppo software è principio teorico che indica il metodo da seguire nel progettare e nello scrivere un programma. Il modello è alla base di una metodologia di sviluppo. I modelli di sviluppo software simulano la realtà per vedere cosa accadrebbe, e al fine di ridurre gli errori e di ottimizzare le prestazioni e i risultati.

In questo progetto la scelta del modello è stata orientata verso l'implementazione del Modello evolutivo, che si divide nelle seguenti fasi:

- Analisi dei requisiti
- Costruzione del prototipo
- Valutazione del prototipo da parte dell'utente
- Modifica il progetto in funzione delle valutazioni ricevute
- Progetto
- Sviluppo
- Test
- Manutenzione

Nei successivi capitoli vengono spiegate le varie fasi di tale modello ad eccezione della manutenzione dato che il progetto non è ancora arrivato ad una release di produzione.

# Capitolo 3

## Specifica

–Il termine specifica, nell’Ingegneria del Software, viene usato in diversi contesti con significati diversi. In genere si può definire come un accordo tra produttore ed utente.–

La specifica generale consegnataci dal Prof. Roberto Giacobazzi è stata la creazione di uno strumento che abbia la capacità trasformare un’istruzione assembly in una sequenza di sole istruzioni mov non modificandone la semantica.

### 3.1 Requisiti

La stesura dei requisiti utente e sistema non è stata considerata dato che nella nostra situazione non c’è alcun dialogo tra sviluppatori e clienti perchè ci impersonifichiamo in entrambe le figure. I requisiti funzionali e non funzionali da noi analizzati, a partire dalla specifica del progetto sono i seguenti.

#### **Requisiti funzionali**

I requisiti funzionali descrivono le funzionalità del sistema software, in termini di servizi che il sistema software deve fornire, di come il sistema software reagisce a specifici tipi di input e di come si comporta in situazioni particolari.

- Creazione di una piattaforma multiutente.
- Possibilità di caricamento di più malware.
- Offrire il servizio (automatico all’upload) della movinazione del codice malware.
- Download del malware movinato.
- Creazione di un’area di amministrazione degli utenti.
- Creazione di un’area di amministrazione dei malware caricati.

#### **Requisiti non funzionali**

Descrivono le proprietà del sistema software in relazione a determinati servizi o funzioni.

- Utilizzo di Repository GitHub
- Utilizzo del Framework Yii
- Utilizzo di un DBMS MySql
- Password crittografate nel database

# Capitolo 4

## Scenari d'uso

Gli scenari sono esempi reali di come un sistema può essere utilizzato, questi ad esempio possono essere: una descrizione della situazione di partenza, una descrizione del normale flusso di eventi, una descrizione di ciò che può andare storto, informazioni su altre attività concorrenti, una descrizione dello stato quando lo scenario termina.

### 4.1 Use Case Diagram

Gli Use Case Diagram descrivono il comportamento funzionale del sistema, come visto dall'utente. Il seguente Use Case Diagram è stato progettato tenendo in considerazione tutte le possibili iterazioni col sistema da parte degli attori Guest, Utente e Admin.

Le scede di specifica degli Use Case invece sono state progettate per cinque possibili azioni:

- Registrazione al sistema da parte dell'attore Guest
- Upload Malware da parte dell'attore Utente
- Download Malware da parte dell'attore Utente
- Visualizzazione della lista utenti da parte dell'attore Admin
- Delete di un Malware da parte dell'attore Admin.

### 4.1.1 Attore Guest

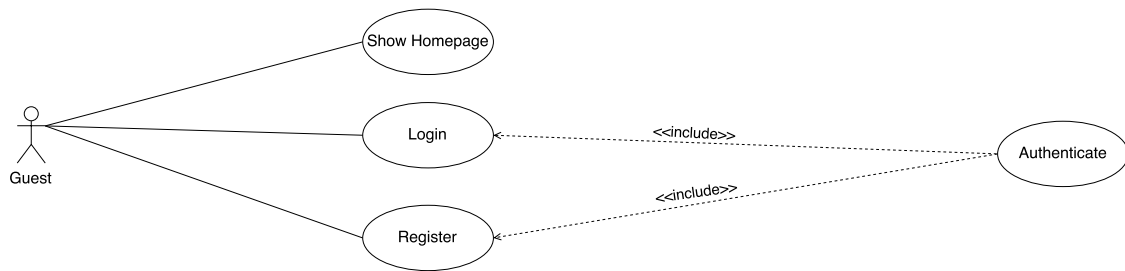


Figura 4.1: Use Case Diagram per l'attore Guest

Caso d'uso: Registrazione Utente
ID: UC2
Attori: Guest
Precondizioni: 1. L'utente non deve essere registrato all'applicazione
Sequenza degli eventi: 1. Il caso d'uso comincia quando l'utente avvia una procedura di registrazione all'applicazione 2. L'utente compila i campi richiesti dall'applicazione e invia la richiesta 3. L'utente invia i dati all'applicazione
PostCondizioni: 1. L'utente deve essere in grado di autenticarsi all'applicazione

Caso d'uso: Login Guest
ID: UC6
Attori: Guest
Precondizioni: 1. L'utente deve essere correttamente registrato all'applicazione
Sequenza degli eventi: 1. Il caso d'uso comincia una volta che l'utente accede alla pagina di login 2. L'utente inserisce i propri dati all'interno degli appositi campi username e password 3. Il sistema autentica l'utente guest
PostCondizioni: 1. L'utente guest deve essere in grado di accedere alle pagine di upload malware e manage malware



### 4.1.2 Attore User

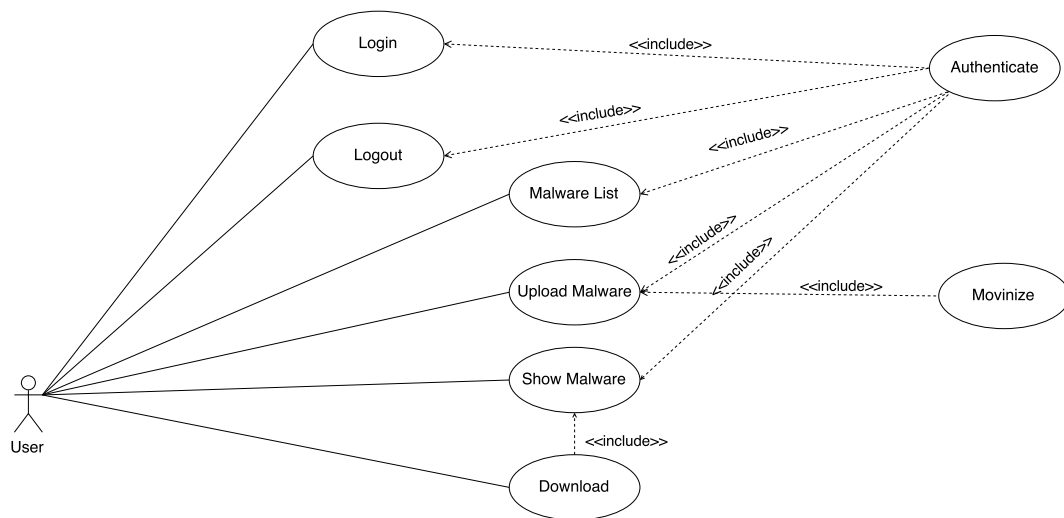


Figura 4.2: Use Case Diagram per l'attore User

Caso d'uso: Upload Malware (Utente)
ID: UC1
Attori: Utente
Precondizioni:
1. L'utente deve essersi autenticato correttamente all'applicazione
Sequenza degli eventi:
1. Il caso d'uso comincia una volta che l'utente si è autenticato correttamente
2. L'utente inserisce all'interno degli appositi campi, Titolo, codice sorgente e invia il tutto
3. Il sistema trasmette un nuovo malware alla base di dati
PostCondizioni:
1. L'utente deve essere in grado di caricare altri malware

Caso d'uso: Download Malware (Utente)
ID: UC3
Attori: Utente
Precondizioni:
1. L'utente deve essere correttamente autenticato all'applicazione
1. Il caso d'uso comincia quando l'utente è correttamente autenticato all'applicazione e avvia la procedura di visualizzazione dei malware salvati sulla base di dati
2. L'utente richiede il download di uno specifico malware
3. Il sistema procede con l'invio di tale file all'utente
PostCondizioni:
1. L'utente deve essere in grado di leggere il contenuto di tale file

### 4.1.3 Attore Admin

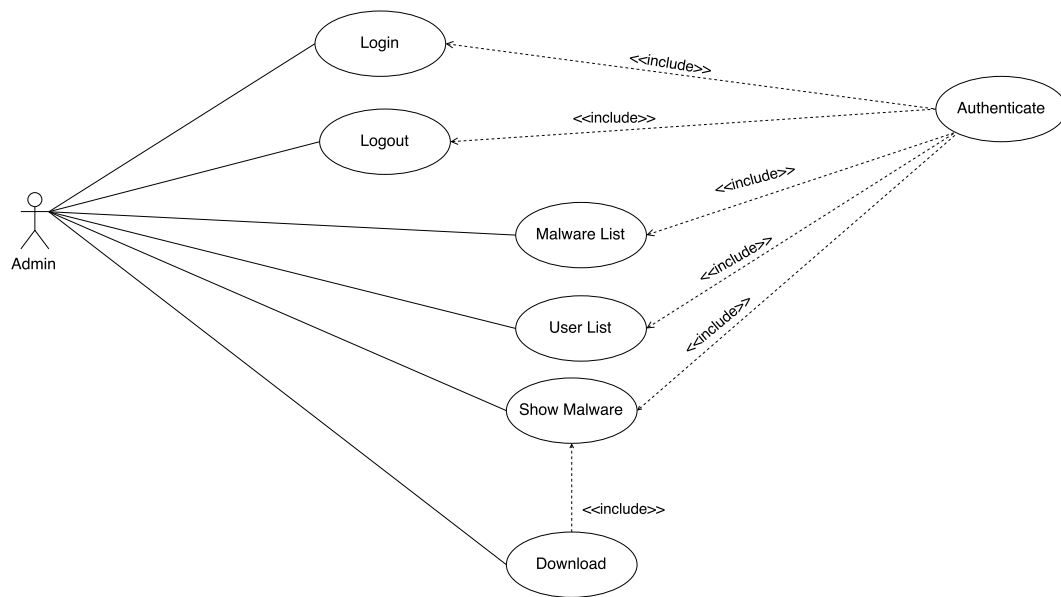


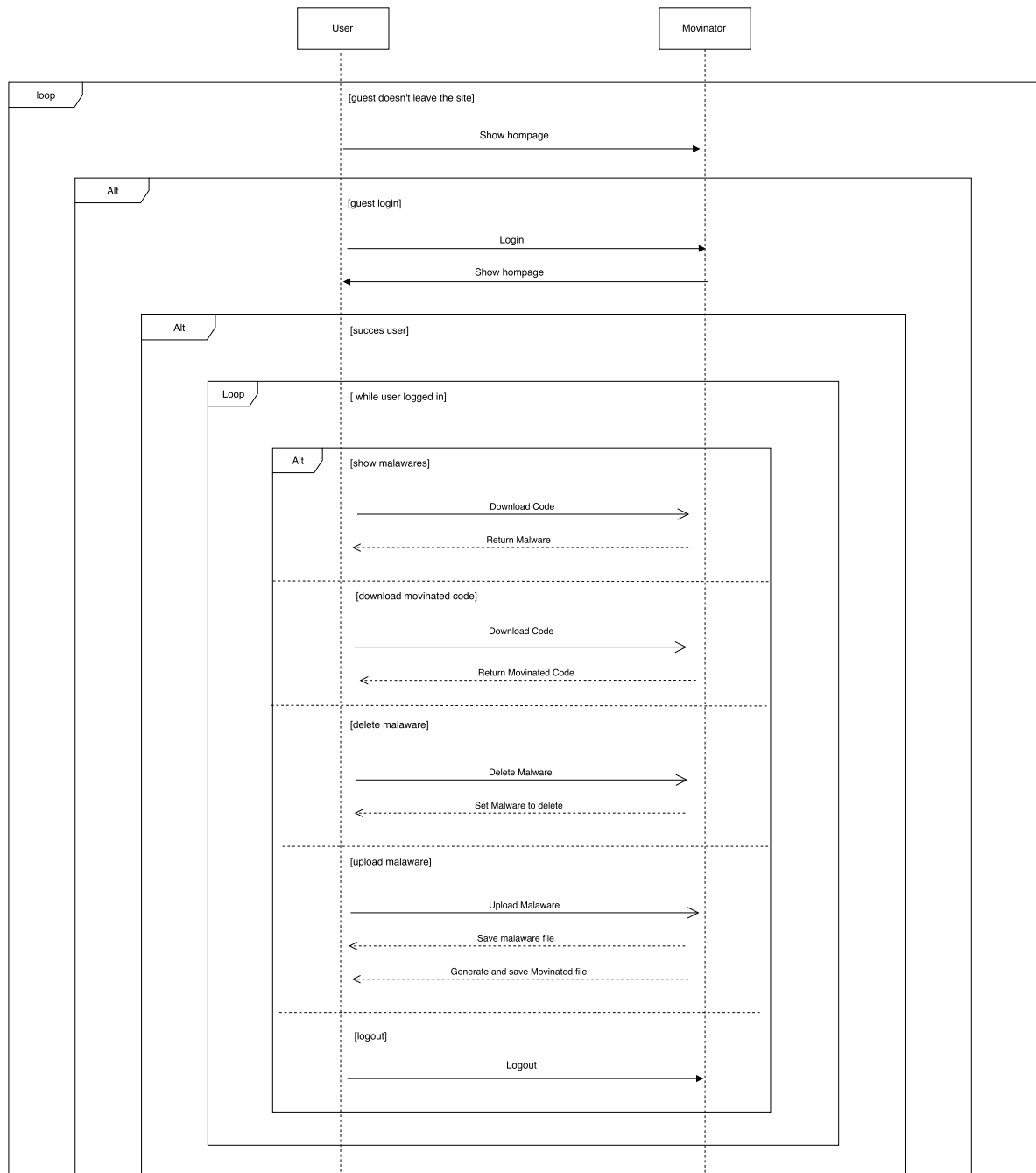
Figura 4.3: Use Case Diagram per l'attore Admin

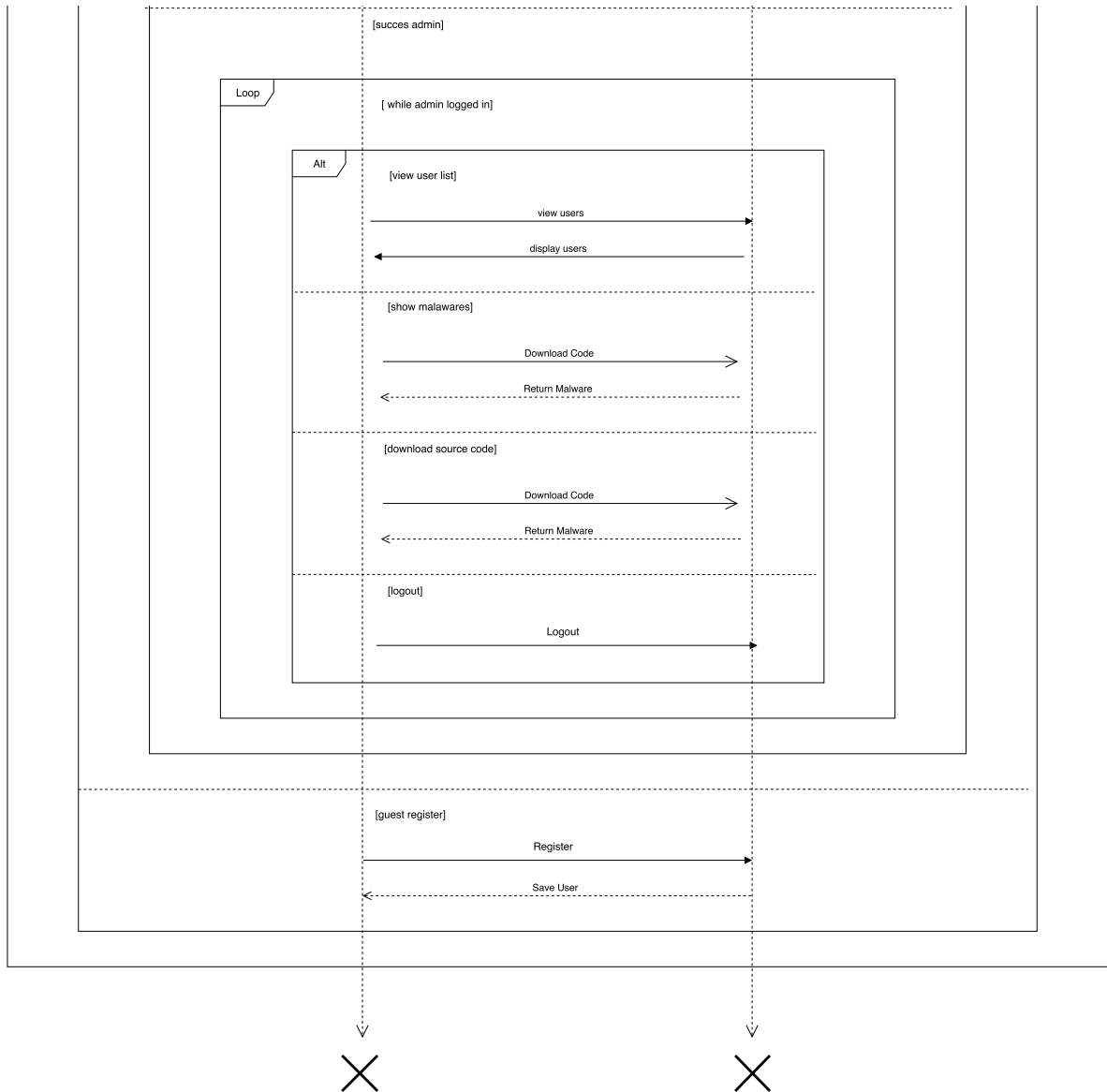
Caso d'uso: Gestione Utenti
ID:UC4
Attori: Admin
Precondizioni:
1. L'amministratore deve essere correttamente autenticato all'applicazione
1. Il caso d'uso comincia quando l'amministratore accede alla pagina della gestione degli utenti
2. Il sistema deve provvedere a visualizzare all'amministratore la pagina con alcune delle specifiche di tutti gli utenti
PostCondizioni:
1. L'amministratore deve essere in grado di visualizzare tutti gli utenti correttamente registrati sulla base di dati

Caso d'uso: Download Malware (Utente)
ID: UC5
Attori: Admin
Precondizioni:
1. L'utente deve essere correttamente autenticato all'applicazione
1. Il caso d'uso comincia quando l'utente è correttamente autenticato all'applicazione e avvia la procedura di visualizzazione dei malware salvati sulla base di dati
2. L'utente richiede il download di uno specifico malware
3. Il sistema procede con l'invio di tale file all'utente
PostCondizioni:
1.L'utente deve essere in grado di leggere il contenuto di tale file

## 4.2 Sequence Diagram degli use case

Il Sequence Diagram è un diagramma di comportamento che esplicita le varie interazioni tra le entità, la rappresentazione viene eseguita dal punto di vista temporale ed è utilizzabile a vari livelli di astrazione: in questo caso è utilizzato a livello degli Use Case.





## Capitolo 5

# Progettazione

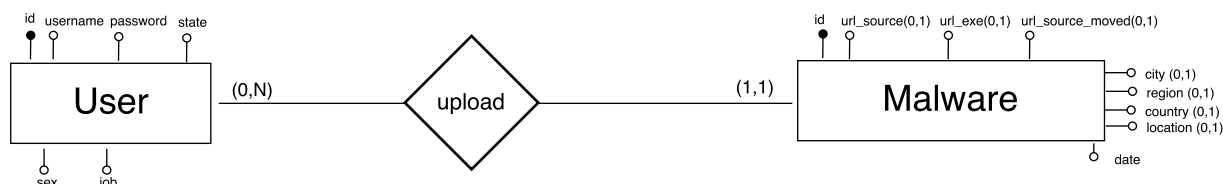
### 5.1 Progettazione Base di Dati

Dato che il ciclo di vita del processo di automazione di un sistema informativo dice che la progettazione del sistema è composto prima dalla progettazione dei dati e poi dalla progettazione delle applicazioni abbiamo descritto qui di seguito la nostra progettazione dei dati che include la progettazione concettuale e la progettazione logica che si concretizzano con la creazione dello schema concettuale e dello schema logico.

Non sono stati applicati nello specifico strategie di progetto (top-down, bottom up, inside-out) vista la compattezza della base di dati.

#### 5.1.1 Schema Concettuale

L'obiettivo della progettazione concettuale il contenuto informativo della base di dati in modo formale ma indipendente dall'implementazione (quindi indipendente dalla scelta del DBMS) e dalle operazioni.



#### 5.1.2 Schema Logico

L'obiettivo della progettazione logica è tradurre lo schema concettuale nello schema logico aderente al modello dei dati del DBMS scelto per l'implementazione. Nella traduzione si tiene conto delle operazioni più frequenti che le applicazione eseguiranno sulla base di dati.

User(id ,username , password, state, sex, job)

Malware(id , userId, title, url\_source\*, url\_exe\*, url\_source\_moved\*, city\*, region\*, country\*, location\*, date)

#### 5.1.3 Schema Fisico

L'obiettivo della progettazione fisica è completare lo schema logico con i parametri relativi alla memorizzazione fisica dei dati e con gli opportuni metodi d'accesso (INDICI) per garantire un accesso efficiente ai dati.

— Database: 'movinator'

---

— Table structure for table 'malware'

```
CREATE TABLE 'malware' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'user_id' int(11) NOT NULL,  
  'title' varchar(128) NOT NULL,  
  'url_exe' varchar(128) DEFAULT NULL,  
  'url_source' varchar(128) DEFAULT NULL,  
  'url_source_moved' varchar(128) DEFAULT NULL,  
  'city' varchar(128) DEFAULT NULL,  
  'region' varchar(128) DEFAULT NULL,  
  'country' int(128) DEFAULT NULL,  
  'location' varchar(128) DEFAULT NULL,  
  'date' date NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

---

— Table structure for table 'user'

```
CREATE TABLE 'user' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'username' varchar(128) NOT NULL,  
  'password' varchar(128) NOT NULL,  
  'email' varchar(128) NOT NULL,  
  'country' varchar(32) NOT NULL,  
  'sex' set('m','f') NOT NULL,  
  'job' set('black hat','white hat','others') NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

— Indexes for table 'malware'

```
ALTER TABLE 'malware'  
  ADD PRIMARY KEY ('id'),  
  ADD KEY 'user_id' ('user_id');
```

— Indexes for table 'user'

```
ALTER TABLE 'user'  
  ADD PRIMARY KEY ('id');
```

— Constraints for table 'malware'

```
ALTER TABLE 'malware'  
  ADD CONSTRAINT 'fk_movinator_id' FOREIGN KEY ('user_id') REFERENCES 'user' ('id')  
  ON DELETE CASCADE ON UPDATE CASCADE;
```

## 5.2 Class Diagram

Il più diffuso diagramma compreso in UML è il diagramma delle classi. Si tratta di un diagramma statico che può essere utilizzato:

- Per la modellazione concettuale del dominio di un problema
- Per la modellazione delle specifiche richieste ad un sistema
- Per modellare l'implementazione (object-oriented) di un sistema software

I concetti fondamentali di un class diagram sono estensioni dei concetti fondamentali dei paradigmi object-oriented. I Principali elementi dei class diagram sono: classi (rappresentanti i tipi di dati presenti in un sistema), associazioni (rappresentano i collegamenti fra istanze di classi), attributi (dati semplici presenti nelle classi e nelle loro istanze), operazioni (rappresentano le funzioni svolte dalle classi e dalle loro istanze) e generalizzazioni (raggruppano le classi in gerarchie di ereditarietà)

I class diagram sono stati progettati prendendo in considerazione le tre maggiori entità del nostro progetto:

- Site: Questa entità comprende la parte di autenticazione dell'applicazione e delle viste iniziali da parte di un utente (home, login, registrazione e index varie per ogni tipologia di utente loggato)
- Malware: Questa entità si occupa dell'iterazione tra l'utente e l'applicazione memorizzando i malware caricati e generandone il corrispondente file Movinato
- User: Quest'ultima entità comprende i metodi per la gestione degli utenti e per la visualizzazione degli stessi da parte dell'amministratore

### 5.2.1 Site

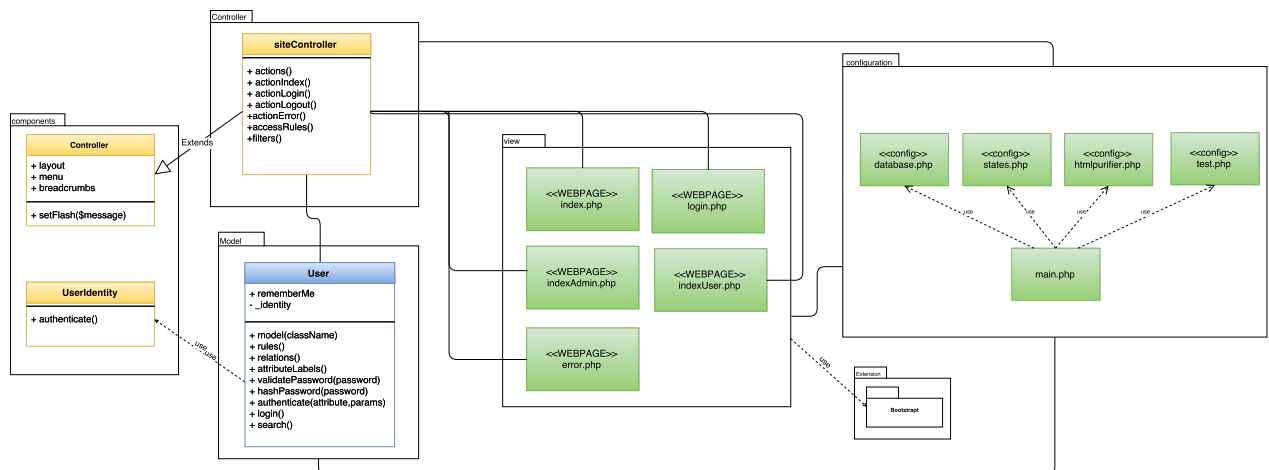


Figura 5.1: Class diagram per l'entità Site

## 5.2.2 Malware

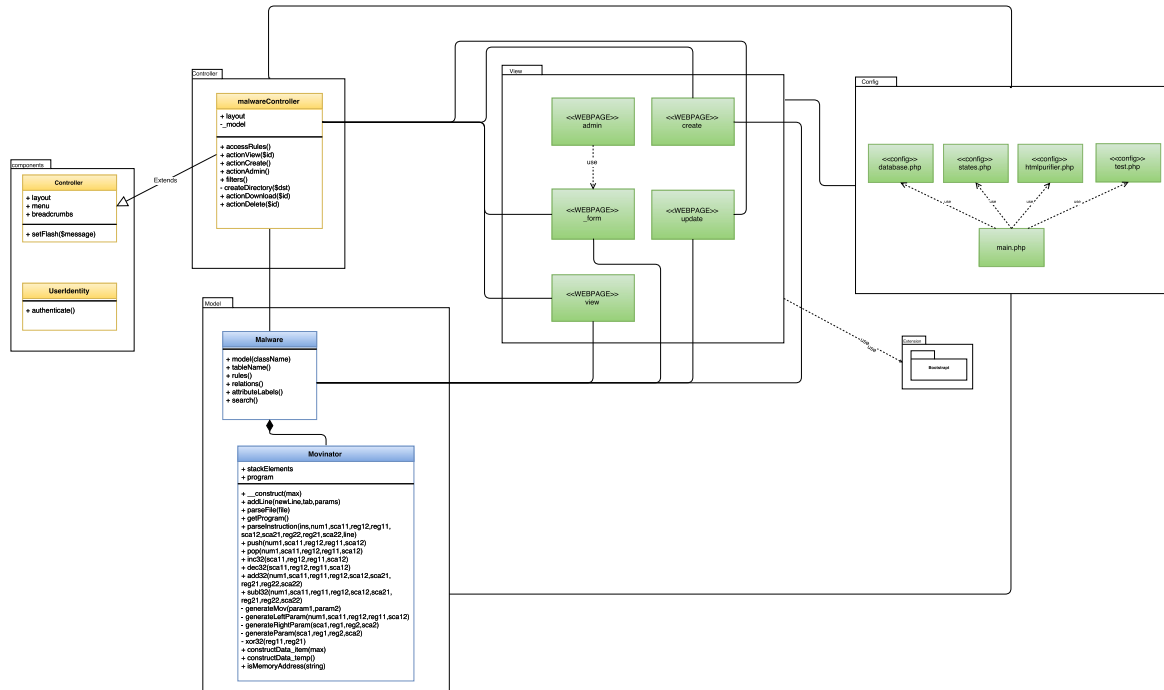


Figura 5.2: Class diagram per l'entità Malware

## 5.2.3 User

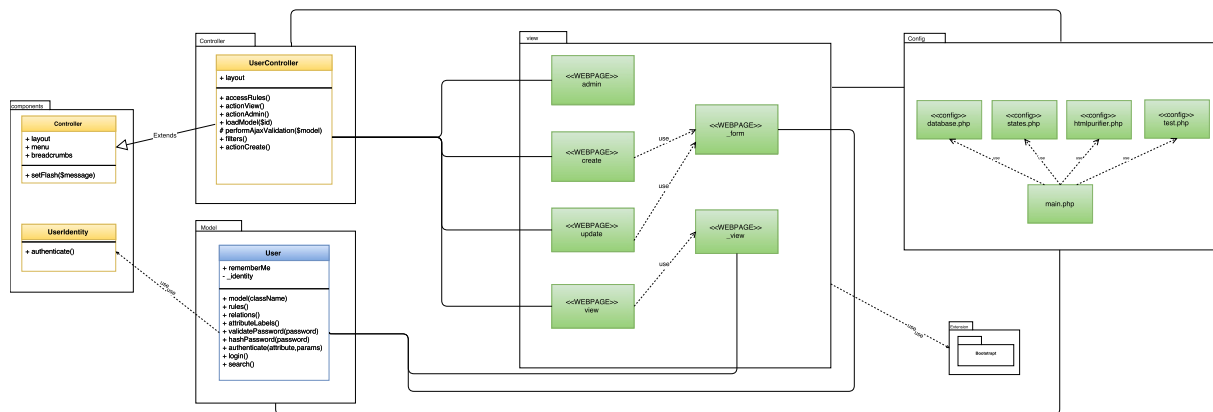
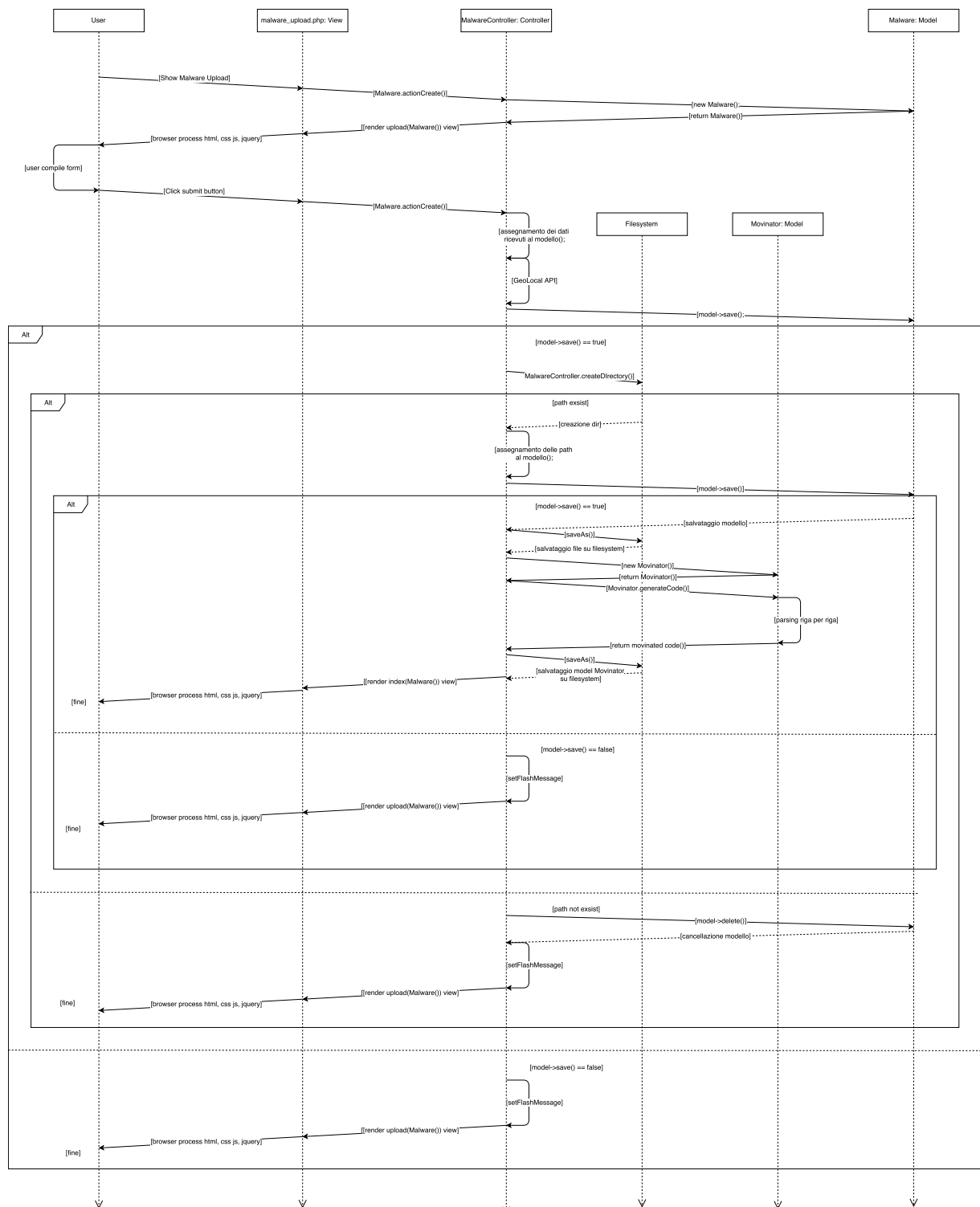


Figura 5.3: Class diagram per l'entità User



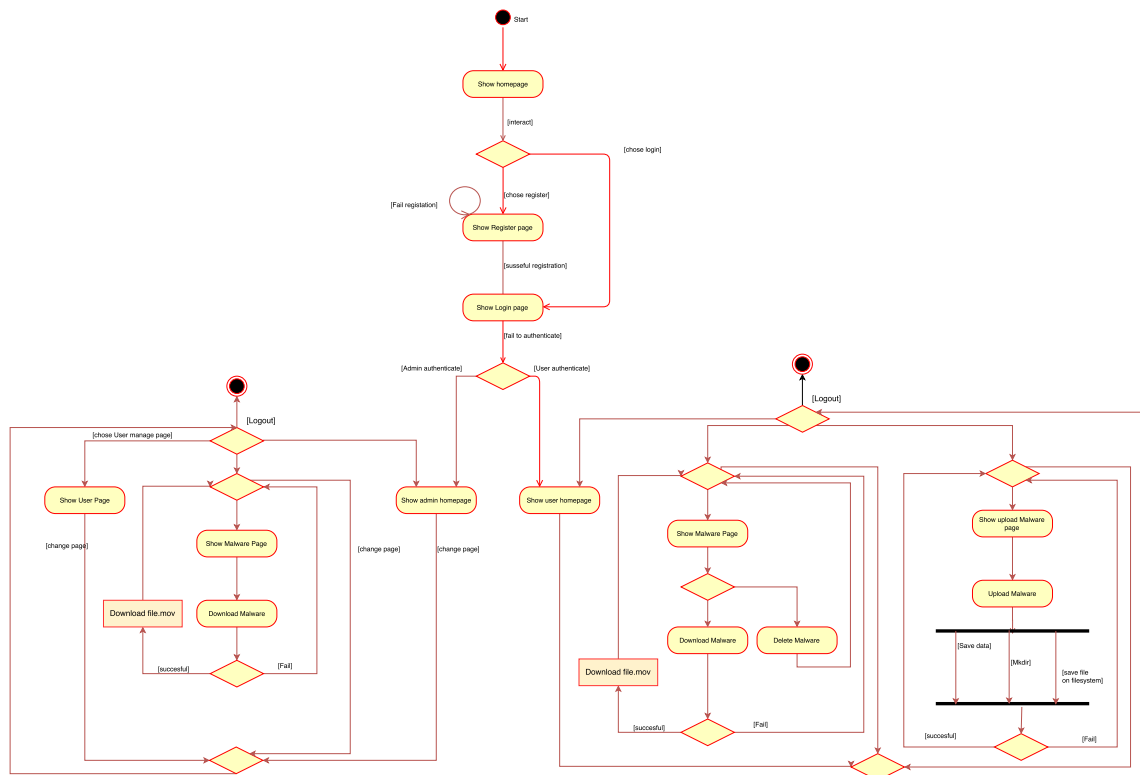
## 5.3 Sequence Diagram

Sono diagrammi di comportamento che modellano le interazioni tra varie entità di un sistema. Visualizzano lo scambio di messaggi tra entità nel tempo. Il loro scopo è mostrare come un certo comportamento viene realizzato dalla collaborazione delle entità in gioco.



## 5.4 Activity Diagram

L'Activity Diagram è un diagramma definito all'interno dello Unified Modeling Language (UML) che definisce le attività da svolgere per realizzare una data funzionalità. Può essere utilizzato durante la progettazione del software per dettagliare un determinato algoritmo. Più in dettaglio, un activity diagram definisce una serie di attività o flusso, anche in termini di relazioni tra le attività, i responsabili per le singole attività e i punti di decisione. L'activity diagram è spesso usato come modello complementare allo Use Case Diagram, per descrivere le dinamiche con cui si sviluppano i diversi use case.



## Capitolo 6

# Architettura

### 6.1 Lamp

LAMP è un acronimo di Linux Apache Mysql Php. Indica una piattaforma software per ambienti server, nella quale sono installati i componenti sopracitati. Una caratteristica che accomuna tutti questi componenti è la loro licenza; sono tutti freeware e software Open Source coperti da licenza GPL. Vorrei precisare inoltre che i software Open Source non sono inefficienti e poco sicuri perché gratis, ma anzi garantiscono un livello di efficienza, affidabilità e sicurezza molto elevato essendo progettati e soprattutto testati da una comunità molto vasta. Esempi di Server LAMP, infatti, sono quelli in uso da Wikipedia, Mozilla e Facebook.

I componenti dettagliati della piattaforma che utilizziamo sono:

- Linux
- Apache, un web server sviluppato dalla Apache Software Foundation che attualmente è il web server più diffuso. Il server Web è un programma che ha il compito di ascoltare eventuali richieste, solitamente fatte sulla porta 80 (HTTP) e 443 (HTTPS), da soddisfare.
- MySQL, un RDBMS, relational database management system. Un RDBMS è un sistema di gestione di basi di dati relazionali. Esso si occupa di gestire la struttura dei dati sul server e di rispondere alle interrogazioni che un applicazione gli pone secondo un linguaggio di interrogazione standard SQL.
- PHP, uno dei linguaggi web più diffusi su internet per la realizzazione di applicazioni web dinamiche ed è utilizzato per scrivere applicazioni web lato server.

### 6.2 Yii 1.7 - PHP Framework

Yii è un framework pensato per sviluppare applicazioni web sfruttando il noto linguaggio di programmazione PHP. È open source, e il suo scopo è quello di aiutare gli sviluppatori web a creare potenti applicazioni in breve tempo.

Come tutti i moderni framework web, è basato sull'architettura MVC ed è integrato con un framework JavaScript altrettanto noto come JQuery con il quale è possibile creare potenti applicazioni AJAX.

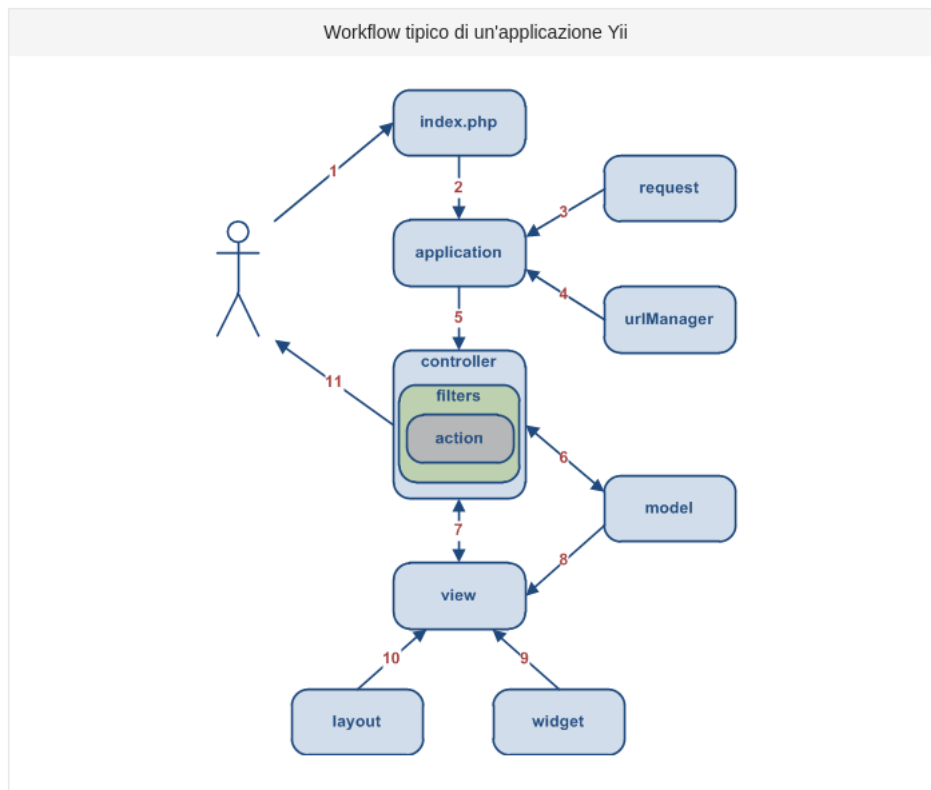


Figura 6.1: Generico flusso di dati in Yii

- Un utente esegue una richiesta all'URL `http://www.example.com/index.php?r=post/show&id=1` ed il web server gestisce la richiesta eseguendo lo script di avvio `index.php`.
- Lo script di avvio crea un'istanza dell'Applicazione e la esegue.
- L'Application riceve le informazioni dettagliate della richiesta dell'utente da un component dell'applicazione che si chiama `request`.
- L'Application determina sia il controller che l'action richiesti grazie all'aiuto dell'application component che si chiama `urlManager`. In questo esempio, il controller è `post`, il quale si riferisce alla classe `PostController`; e l'action è `show`, il cui significato attuale è determinato dal controller.
- L'applicazione crea una istanza del controller richiesto per gestire ulteriormente la richiesta utente. Il controller determina che l'action `show` si riferisce al metodo, all'interno della classe del controller, che si chiama `actionShow`. Il metodo crea ed esegue i filtri (es. controllo accessi, benchmark) associati con questa action. L'action viene eseguita se ciò è permesso dai filtri.
- L'action, tramite il model, legge dal database il `Post` il cui ID è 1.
- L'action produce una view chiamata `show` con i prodotti dal model `Post`.
- La view legge e visualizza gli attributi del model `Post`.
- La view esegue alcuni widget.
- La produzione della view viene incorporata in un layout.
- L'action completa la produzione della view e ne visualizza il risultato all'utente.

## 6.3 Repository GitHub

Un sistema di controllo versione distribuito permette di tenere traccia delle modifiche e delle versioni apportate al codice sorgente di un software. Con questo sistema gli sviluppatori possono collaborare individualmente e parallelamente da offline su di un proprio ramo (branch) di sviluppo, registrare le proprie modifiche (commit) ed in seguito condividerle con altri o unite (merge) a quelle di altri.

Come sistema utilizziamo GitHub.com che è un servizio di hosting per progetti software. Il nome deriva dal fatto che GitHub è un servizio sostitutivo del software dell'omonimo strumento di controllo versione distribuito, Git.

Per sincronizzare il repository situato sul Desktop con apache2 abbiamo configurato un virtualhost nel seguente modo:

```
<VirtualHost *:80>
    <Directory "/home/jeko/Desktop/movinator">
        Options FollowSymLinks
        AllowOverride None
        Require all granted
    </Directory>
    DirectoryIndex movinator/index.php
    DocumentRoot "/home/jeko/Desktop/movinator"
    ServerName www.movinator.com
</VirtualHost>
```

## 6.4 Qualità Architetture

Per contribuire in maniera adeguata alla soluzione del problema applicativo illustriamo qui sotto le qualità che l'architettura deve possedere, specificando ad alto livello quali saranno i processi che garantiranno la qualità descritta.

- **Performace:** Il sistema già reagisce adeguatamente rispetto al carico di lavoro atteso. L'unico punto critico potrebbe essere il caricamento di un malware, ad esempio di 1GB. Per ovviare a questo problema verranno introdotti dei processi per l'incremento dell'User experience design in caso di lunghe attese.
- **Scalabilità:** Il sistema sarà in grado di adattarsi a situazioni di carico superiori a quelle previste inizialmente inserendo dei balancer su più livelli. I punti critici di scalabilità potrebbero essere a livello di database e di filesystem. In entrambi i casi si possono adottare soluzioni che bilancino il carico di lavoro nel caso del database utilizzando dei cluster mysql e nel caso del filesystem adottare delle soluzioni RAID per la memoria secondaria del server che ospita l'applicazione. Queste ultime due soluzioni incrementano anche la disponibilità del servizio.
- **Sicurezza**
- **Affidabilità** Per quanto ci è stato possibile abbiamo implementato l'architettura in modo da limitare le possibilità di malfunzionamento dell'applicazione.

**Disponibilità:** Il sistema rende disponibile a ciascun utente abilitato le informazioni alle quali ha diritto di accedere, nei tempi e nei modi previsti. Per ora non ci sono ovvi problemi di disponibilità visto che l'applicazione verrà ospitata da un provider hosting, in caso di malfunzionamenti non abbiamo strumenti di riparo non avendo fisicamente il server a nostra disposizione.

**Integrità:** L'applicazione idealmente impedisce l'alterazione diretta o indiretta delle informazioni sia da parte di utenti e processi non autorizzati che a seguito di eventi accidentali. Questa qualità può essere estesa sia ai dati che al sistema.

**Confidenzialità:** Il sistema non permette ad un utente di poter ottenere o dedurre dal sistema informazioni che non è autorizzato a conoscere garantendo privacy e riservatezza dei dati.

# Capitolo 7

## Sviluppo

### 7.1 Realizzazione prototipo

Un prototipo è un modello approssimato o parziale del sistema che vogliamo sviluppare che simula o esegue alcune funzioni del sistema finale, realizzato allo scopo di valutarne le caratteristiche, in particolare l'usabilità.

Il prototipo di Movinator è stato progettato e sviluppato in linguaggio Java con l'utilizzo di uno script Bash per la velocità nell'implementazione dei controlli. È stato consegnato quindi, al Prof. Giacobazzi un PoC (Proof of Concept) cioè un abbozzo del progetto, con lo scopo di dimostrarne la fattibilità o la fondatezza dei principi e concetti che costituivano il progetto. Questo PoC conteneva il wrapper Bash, il core di Movinator scritto in Java ed una serie di test costituiti da un coppie di codice sorgente assembly e sorgente dello stesso programma movinato.

Il prototipo in questo caso, ha avuto il ruolo di "Studio di fattibilità" sulla reale concretizzazione dell'idea originale del Prof. Giacobazzi essendo una cosa che non è stata implementata da nessun'altra persona. Da qui si è scelto di sviluppare un SaaS che metta a disposizione l'upload e la gestione dei malware da parte degli utenti e l'utilizzo del servizio di Movinazione.

### 7.2 Realizzazione SaaS

Il modello SaaS è identificato anche come un modello di software on demand(Cloud Computing) distribuito come servizio in hosting; si tratta di un modello di cloud in cui all'utente viene messo a disposizione direttamente un ambiente completo con l'applicazione da utilizzare, sollevandolo dalla responsabilità e dall'onore dell'installazione dell'applicazione, della sua configurazione, e della configurazione del sistema host che eseguirà l'applicazione.

## Capitolo 8

# Test sul prototipo

### 8.1 Push

L'istruzione Push si occupa di trasferire nello stack il dato specificato come parametro. Movinator dopo aver trasferito il dato non decrementa il valore del registro `%esp`, ma mantiene un contatore all'interno che memorizza il numero dei valori inseriti sullo stack. Grazie a questo valore riusciamo ad inserire gli elementi nella corretta posizione usando `%esp` come registro base, sommato al numero degli elementi presenti sullo stack moltiplicato per la grandezza dell'indirizzamento dell'architettura (4 Byte per 32 bit).

#### 8.1.1 Push intero

<code>push \$4, %ebx</code>	<code>movl \$4, num(%esp)</code>
-----------------------------	----------------------------------

`movl $4, num(%esp)`

Viene inserito il numero specificato, in questo caso 4, sullo stack. Per **num** si intende l'n+1-esimo valore che è stato inserito sullo stack.

#### 8.1.2 Push registro

<code>push %eax</code>	<code>movl %eax, num(%esp)</code>
------------------------	-----------------------------------

`movl %eax, num(%esp)`

Viene inserito il valore del registro specificato, in questo caso `%eax`, sullo stack. Per **num** si intende l'n+1-esimo valore che è stato inserito sullo stack.

### 8.1.3 Push memoria

<code>push 512(%ebx,%edx,4)</code>	<code>movl 512(%ebx,%edx,4) , num(%esp)</code>
------------------------------------	--

`movl 512(%ebx,%edx,4), num(%esp)` Viene inserito il valore di memoria specificato, in questo caso `512(%ebx,%edx,4)`, sullo stack. Per ***num*** si intende l' $n+1$ -esimo valore che è stato inserito sullo stack.



## 8.2 Pop

Preleva un dato dallo stack e lo memorizza in uno specifico registro o variabile. L'istruzione Pop si occupa di prelevare dallo stack l'ultimo valore inserito memorizzandolo nel registro o nella variabile specificata come parametro. Movinator dopo aver trasferito il dato non incrementa il registro %esp, ma mantiene un contatore all'interno che memorizza il numero dei valori inseriti sullo stack. Grazie a questo valore riusciamo a prelevare gli elementi nella corretta posizione usando %esp come registro base, sommato al numero degli elementi presenti sullo stack moltiplicato per la grandezza dell'indirizzamento dell'architettura (4 Byte per 32 bit).

### 8.2.1 Pop variabile

<code>pop var</code>	<code>movl num(%esp), var</code>
----------------------	----------------------------------

`movl num(%esp), var`

Viene inserito nella variabile specificata l'ultimo valore inserito sullo stack. Per **num** si intende l'n-esimo valore che è stato inserito sullo stack.

### 8.2.2 Pop registro

<code>pop %eax</code>	<code>movl num(%esp), %eax</code>
-----------------------	-----------------------------------

`movl num(%esp), %eax`

Viene inserito nel registro specificato, in questo caso 512(%ebx,%edx,4), l'ultimo valore inserito sullo stack. Per **num** si intende l'n-esimo valore che è stato inserito sullo stack.

### 8.2.3 Pop memoria

<code>pop 512(%ebx,%edx,4)</code>	<code>movl num(%esp), 512(%ebx,%edx,4)</code>
-----------------------------------	---

`movl 512(%ebx,%edx,4), num(%esp)` Viene inserito nella variabile specificata l'ultimo valore inserito sullo stack. Per **num** si intende l'n-esimo valore che è stato inserito sullo stack.

## 8.3 Inc

Istruzione che ha una sintassi simile alla push e alla pop con un unico parametro che può essere un registro o una locazione di memoria. Permette di incrementare di 1 il valore dell'operando specificato.

### 8.3.1 Inc registro

<code>inc %eax</code>	<pre>movl %edx, temp movl \$4, %edx movl data_items+512(%edx, %eax, 4), %eax movl temp, %edx</pre>
-----------------------	--

<code>movl %edx, temp</code>	Viene salvato nella variabile temp il valore contenuto nel registro %edx
<code>movl \$4, %edx</code>	Viene inserito all'interno di %edx la costante 4 che rappresenta il numero di bit che dobbiamo avanzare per ottenere il successivo valore all'interno del nostro array locazione-valore
<code>movl data_items+512(%edx, %eax, 4), %eax</code>	Viene salvato in %eax il valore successivo ottenuto sommando 4 bit al valore puntato dalla locazione di memoria %eax * 4
<code>movl temp, %edx</code>	Viene ripristinato il valore precedentemente contenuto nel registro %edx

### 8.3.2 Inc memoria

<code>inc 4(%esp,%eax,4)</code>	<pre> movl %edx, temp movl %ecx, temp2 movl \$4, %edx movl 4(%esp, %eax, 4), %ecx movl data_items+512(%edx, %ecx, 4), %ecx movl %ecx, 4(%esp, %eax, 4) movl temp, %edx movl temp2, %ecx </pre>
---------------------------------	--

<code>movl %edx, temp</code>	Viene salvato nella variabile temp il valore contenuto nel registro %edx
<code>movl %ecx, temp2</code>	Viene salvato nella variabile temp il valore contenuto nel registro %ecx
<code>movl \$4, %edx</code>	Viene salvato all' interno di %edx la costante 4 che rappresenta il numero di bit che dobbiamo avanzare per ottenere il successivo valore all' interno del nostro array locazione-valore
<code>movl 4(%esp, %eax, 4), %ecx</code>	Viene salvato in %ecx il valore puntato dalla locazione di memoria puntata da $4 + \%esp + \%eax * 4$ che poi rappresenterà la posizione base sul nostro array locazione-risultato
<code>movl data_items+512(%edx, %ecx, 4), %ecx</code>	Viene salvato in %eax il valore successivo ottenuto sommando 4 bit al valore puntato dalla locazione di memoria $\%eax * 4$
<code>movl %ecx, 4(%esp, %eax, 4)</code>	Viene salvato il risultato precedentemente ottenuto dall' incremento all' interno della locazione di destinazione dell' operazione
<code>movl temp, %edx</code>	Viene ripristinato il valore precedentemente contenuto in %edx
<code>movl temp2, %ecx</code>	Viene ripristinato il valore precedentemente contenuto in %ecx

## 8.4 Dec

Istruzione che ha una sintassi simile alla push e alla pop con un unico parametro che può essere un registro o una locazione di memoria. Permette di decrementare di 1 il valore dell'operando specificato.

### 8.4.1 Dec registro

<code>dec %eax</code>	<pre>movl %edx, temp movl \$-4, %edx movl data_items+512(%edx, %eax, 4), %eax movl temp, %edx</pre>
-----------------------	---

<code>movl %edx, temp</code>	Viene salvato nella variabile temp il valore contenuto nel registro %edx.
<code>movl \$-4, %edx</code>	Viene salvato all'interno di %edx la costante -4 che rappresenta il numero di bit che dobbiamo sottrarre per ottenere il precedente valore all'interno del nostro array locazione-valore.
<code>movl data_items+512(%edx, %eax, 4), %eax</code>	Viene salvato in %eax il valore successivo ottenuto sottraendo 4 bit al valore puntato dalla locazione di memoria %eax * 4.
<code>movl temp, %edx</code>	Viene ripristinato il valore precedentemente contenuto nel registro %edx.

### 8.4.2 Dec memoria

<code>dec 4(%esp,%eax,4)</code>	<pre>movl %edx, temp movl %ecx, temp2 movl \$-4, %edx movl 4(%esp, %eax, 4), %ecx movl data_items+512(%edx, %ecx, 4), %ecx movl %ecx, 4(%esp, %eax, 4) movl temp, %edx movl temp2, %ecx</pre>
---------------------------------	---

`movl %edx, temp`

Viene salvato nella variabile temp il valore contenuto nel registro %edx.

`movl %ecx, temp2`

Viene salvato nella variabile temp il valore preventivamente contenuto nel registro %ecx.

`movl $-4, %edx`

Viene salvato all' interno di %edx la costante -4 che rappresenta il numero di bit che dobbiamo sottrarre per ottenere il precedente valore all' interno del nostro array locazione-valore.

`movl 4(%esp, %eax, 4), %ecx`

Viene salvato in %ecx il valore puntato dalla locazione di memoria puntata da  $4 + \%esp + \%eax * 4$  che poi rappresenterà la posizione base sul nostro array locazione-risultato.

`movl data_items+512(%edx, %ecx, 4), %ecx`

Viene salvato in %ecx il valore successivo ottenuto sottraendo 4 bit al valore puntato dalla locazione di memoria %eax \* 4.

`movl %ecx, 4(%esp, %eax, 4)`

Viene salvato il risultato precedentemente ottenuto dal decremento all' interno della locazione di destinazione dell' operazione.

`movl temp, %edx`

Viene ripristinato il valore precedentemente contenuto in %edx.

`movl temp2, %ecx`

Viene ripristinato il valore precedentemente contenuto in %ecx.

## 8.5 Add

La Add è un'istruzione aritmetica che prende in input due operandi, una sorgente e una destinazione. Ritornando la somma nel registro di destinazione. La destinazione deve essere un registro o una locazione di memoria. Mentre la sorgente può essere una locazione di memoria, un registro o una costante. Da notare che le operazioni tra locazioni di memoria non sono possibili nel linguaggio assembly, quindi almeno uno dei due deve essere un registro o una costante.

### 8.5.1 Add intero, registro

<code>addl \$4,%ebx</code>	<pre>movl %edx,temp movl data_items+512(,%ebx,8),%ebx movl data_items+512(,%ebx,8),%ebx movl \$4,%edx movl data_items+512(%ebx,%edx,4),%ebx movl temp,%edx</pre>
----------------------------	--

<code>movl %edx,temp</code>	Viene salvato preventivamente all'interno di una variabile il valore di %ecx, in modo tale da non perderlo, evitando così perdita di informazioni.
<code>movl data_items+512(,%ebx,8),%ebx</code>	Serve per recuperare da data_items la cella di memoria puntata dal valore contenuto in %ebx.
<code>movl \$4,%edx</code>	Viene salvato all'interno di %edx il valore 4, per usarla poi per lo spostamento all'interno di data_items.
<code>movl data_items+512(%ebx,%edx,4),%ebx</code>	Passaggio fondamentale del paper, implementazione della somma tramite spostamento in memoria. Andiamo a salvare nel registro di destinazione il contenuto della cella di memoria del nostro array locazione-risultato data_items alla posizione $\%ebx + \%edx * 4$ . Il nostro scalare è 4 perché stiamo lavorando con un array di tipo long(4 byte).
<code>movl temp,%edx</code>	Viene ripristinato il precedente valore di %edx.

## 8.5.2 Add intero , memoria

addl \$4,4(%esp,%ebx,4)	<pre> movl %edx,temp movl %ecx,temp2 movl 4(%esp,%ebx,4),%edx movl data_items+512(,%edx,8),%edx movl data_items+512(,%edx,8),%edx movl \$4,%ecx movl data_items+512(%edx,%ecx,4),%ebx movl temp,%edx movl temp2,%ecx </pre>
-------------------------	---

movl %edx, temp	Viene salvato preventivamente all' interno di una variabile il valore di %ecx, in modo tale da non perderlo, evitando così perdita di informazioni.
movl %ecx, temp2	Viene salvato preventivamente all' interno di una variabile il valore di %ecx, in modo tale da non perderlo, evitando così perdita di informazioni.
movl 4(%esp, %ebx, 4), %edx	Viene salvato il valore in memoria puntato da (%esp + %ebx * 4) + 4 per poi utilizzarlo come registro di destinazione nella nostra operazione.
movl data_items+512(,%edx,8), %edx	Viene usato per recuperare da data_items la cella di memoria puntata dal valore contenuto in %edx.
movl \$4, %ecx	Viene salvato all' interno di %ecx il valore 4, per usarla poi per lo spostamento all ' interno di data_items.
movl data_items+512(%edx,%ecx,4), %edx	<p>Passaggio fondamentale del paper, implementazione della somma tramite spostamento in memoria. Andiamo a salvare nel registro di destinazione il contenuto della cella di memoria del nostro array locazione-risultato data_items alla posizione %edx + %ecx * 4.</p> <p>Il nostro scalare é 4 perché stiamo lavorando con un array di tipo long(4 byte).</p>
movl %edx, 4(%esp, %ebx, 4)	Serve a memorizzare il valore dell' operazione alla cella di memoria puntata dall equazione $4 + \%esp + \%ebx * 4$ .
movl temp, %edx	Viene ripristinato il precedente valore di %edx.
movl temp2, %ecx	Viene ripristinato il precedente valore di %ecx.

### 8.5.3 Add registro, registro

<code>addl %eax,%ebx</code>	<code>movl data_items+512(,%ebx,8),%ebx</code> <code>movl data_items+512(,%ebx,8),%ebx</code> <code>movl data_items+512(%ebx,%eax,4),%ebx</code>
-----------------------------	--

`movl data_items+512(,%ebx,8),%ebx` Viene utilizzata per recuperare da `data_items` la cella di memoria puntata dal valore contenuto in `%ebx`

`movl data_items+512(%ebx,%eax,4),%ebx` Passaggio fondamentale del paper, implementazione della somma tramite spostamento in memoria. Serve per andare a recuperare nel registro di destinazione il contenuto della cella di memoria del nostro array locazione-risultato `data_items` alla posizione `%ebx + %eax * 4`. Il nostro scalare è 4 perché stiamo lavorando con un array di tipo `long`(4 byte).



### 8.5.4 Add memoria, registro

<code>addl 4(%esp,%eax,4),%ebx</code>	<code>movl %edx,temp movl data_items+512(,%ebx,8), %ebx movl data_items+512(,%ebx,8), %ebx movl 4(%esp,%eax,4),%edx movl data_items+512(%ebx,%edx,4), %ebx movl temp, %edx</code>
---------------------------------------	---

`movl %edx,temp`

Viene utilizzato per salvare preventivamente all' interno di una variabile il valore di %ecx, in modo tale da non perderlo, evitando così perdita di informazioni.

`movl data_items+512(,%ebx,8), %ebx`

Viene utilizzato per recuperare da data\_items la cella di memoria puntata dal valore contenuto in %ebx.

`movl 4(%esp, %ebx, 4), %edx`

Viene usato per salvare in %edx il valore puntato dalla memoria alla posizione  $4 + (\%esp + \%ebx * 4)$ .

`movl data_items+512(%ebx,%edx,4), %ebx`

Passaggio fondamentale del paper, implementazione della somma tramite spostamento in memoria. Viene utilizzato per salvare nel registro di destinazione il contenuto della cella di memoria del nostro array data\_items alla posizione  $\%ebx + \%edx * 4$ . Il nostro scalare é 4 perché stiamo lavorando con un array di tipo long(4 byte).

`movl temp, %edx`

Viene utilizzata per ripristinare il precedente valore di %edx.

### 8.5.5 Add registro, memoria

<code>addl %eax, 4(%esp,%ebx,4)</code>	<code>movl %edx,temp movl 4(%esp,%ebx,4),%edx movl data_items+512(,%edx,8), %edx movl data_items+512(,%edx,8), %edx movl data_items+512(%ebx,%edx,4), %ebx movl temp, %edx</code>
--	---

`movl %edx, temp`

Viene utilizzata per salvare preventivamente all' interno di una variabile il valore di %edx, in modo tale da non perderlo, evitando così perdita di informazioni.

`movl 4(%esp, %ebx, 4), %edx`

Serve per recuperare il valore in memoria puntato da (`%esp + %ebx * 4`) + 4 per andare a sommarlo con il registro di destinazione.

`movl data_items+512(,%edx,8), %edx`

Le due operazioni sottostanti servono per recuperare da `data_items` la cella di memoria puntata dal valore contenuto in %ebx.

`movl data_items+512(%edx,%eax,4), %edx`

Passaggio fondamentale del paper, implementazione della somma tramite spostamento in memoria. Andiamo a salvare nel registro di destinazione il contenuto della cella di memoria del nostro array `data_items` alla posizione `%ebx + %edx * 4`. Il nostro scalare é 4 perché stiamo lavorando con un array di tipo `long`(4 byte).

`movl %edx, 4(%esp, %ebx, 4)`

Serve per salvare il valore ottenuto dallo spostamento in memoria sul nastro, alla cella di memoria puntata da `4(%esp,%ebx,4)`.

`movl temp, %edx`

Serve per ripristinare il valore contenuto precedentemente in %edx.

## 8.6 Sub

La Sub è un'istruzione aritmetica che prende in input due operandi, una sorgente e una destinazione, ritornando la differenza nel registro di destinazione. La destinazione deve essere un registro o una locazione di memoria, mentre la sorgente può essere una locazione di memoria, un registro o una costante. Da notare che le operazioni tra locazioni di memoria non sono possibili nel linguaggio assembly, quindi almeno uno dei due deve essere un registro o una costante.

### 8.6.1 Sub intero, registro

<code>subl \$3,%ebx</code>	<code>movl data_items+512(,%ebx,8) , %ebx</code> <code>movl data_items+512(,%ebx,8) , %ebx</code> <code>movl %edx, temp</code> <code>movl \$3, %edx</code> <code>movl data_items_negative(,%edx,4) , %edx</code> <code>movl data_items+512(%ebx,%edx,4) , %ebx</code> <code>movl temp, %edx</code>
----------------------------	--

<code>movl data_items+512(,%ebx,8), %ebx</code>	Utilizzato per andare a recuperare la cella di memoria che punta al valore contenuto dal registro %ebx.
<code>movl %edx, temp</code>	Viene salvato in temp il contenuto del registro %edx per evitare la perdita del valore salvato.
<code>movl \$3, %edx</code>	Viene salvata la costante che verrà poi sottratta al valore del registro di destinazione in modo tale da poterla usare come spostamento sul mio array locazione-risultato.
<code>movl data_items_negative(,%edx,4), %edx</code>	Viene recuperato il valore negativo della costante precedentemente salvata in %edx per effettuare la sottrazione.
<code>movl data_items+512(%ebx,%edx,4), %ebx</code>	Parte fondamentale del paper, implementazione della sottrazione tramite l'uso della mov con accesso alla memoria. Viene puntato sul nastro il risultato del registro di destinazione ( %ebx - %edx * 4 ) in modo tale da retrocedere sul nastro di n posizioni pari al valore della costante usata.
<code>movl temp, %edx</code>	Viene ripristinato il valore precedentemente salvato in %edx.

## 8.6.2 Sub intero, memoria

<pre> subl \$3,4(%esp,%ebx,4) </pre>	<pre> movl %edx, temp movl %ecx, temp2 movl 4(%esp, %ebx, 4), %edx movl data_items+512(,%edx,8), %edx movl data_items+512(,%edx,8), %edx movl \$3, %ecx movl data_items_negative(,%ecx,4), %ecx movl data_items+512(%edx,%ecx,4), %edx movl %edx, 4(%esp, %ebx, 4) movl temp, %edx movl temp2, %ecx </pre>
--------------------------------------	--

<code>movl %edx, temp</code>	Viene salvato in temp il contenuto del registro %edx per evitare la perdita del valore salvato.
<code>movl %ecx, temp2</code>	Viene salvato in temp2 il contenuto del registro %ecx per evitare la perdita del valore salvato.
<code>movl 4(%esp, %ebx, 4), %edx</code>	Utilizzato per recuperare il valore puntato dal registro di destinazione. Viene recuperato il valore puntato dalla cella di memoria $4 + \%esp + \%ebx * 4$ per poi effettuare la sottrazione.
<code>movl data_items+512(,%edx,8), %edx</code>	Utilizzato per recuperare la cella di memoria che punta al valore contenuto dal registro %edx.
<code>movl \$3, %ecx</code>	Viene salvata la costante che verrà poi sottratta al valore del registro di destinazione in modo tale da poterla usare come spostamento sul mio array di valori.
<code>movl data_items_negative(,%ecx,4), %ecx</code>	Viene recuperato il valore negativo della costante precedentemente salvata in %ecx per effettuare la sottrazione.
<code>movl data_items+512(%edx,%ecx,4), %edx</code>	Parte fondamentale del paper, implementazione della sottrazione tramite l'uso della mov con accesso alla memoria. Viene puntato sul nostro array il risultato del registro di destinazione $(\%ebx - \%edx * 4)$ in modo tale da retrocedere sull'array di n posizioni pari al valore della costante usata.
<code>movl %edx, 4(%esp, %ebx, 4)</code>	Viene salvato il risultato della sottrazione alla cella puntata dal risultato dell'operazione $4 + \%esp + \%ebx * 4$ .
<code>movl temp, %edx</code>	Viene ripristinato il valore precedentemente contenuto in %edx.
<code>movl temp2, %ecx</code>	Viene ripristinato il valore precedentemente contenuto in %ecx.

### 8.6.3 Sub registro, registro

<code>subl %ebx,%eax</code>	<pre>movl %edx, temp movl %ebx, %edx movl data_items_negative(,%edx,4), %edx movl data_items+512(,%eax,8), %eax movl data_items+512(,%eax,8), %eax movl data_items(%eax,%edx,4), %eax movl temp, %edx</pre>
-----------------------------	---

`movl %edx, temp`

Viene salvato in temp il contenuto del registro %edx per evitare la perdita del valore salvato.

`movl %ebx, %edx`

Viene salvato il valore contenuto nel registro %ebx che poi mi servirà per effettuare l'operazione di spostamento sull'array locazione-risultati.

`movl data_items_negative(,%edx,4), %edx`

Viene recuperato il valore negativo del registro precedentemente salvato in %ecx per effettuare la sottrazione.

`movl data_items+512(,%eax,8), %eax`

Utilizzato per andare a recuperare la cella di memoria che punta al valore contenuto dal registro %eax.

`movl data_items(%eax,%edx,4), %eax`

Parte fondamentale del paper, implementazione della sottrazione tramite l'uso della mov con accesso alla memoria. Viene puntato sul nostro array dei risultati il risultato del registro di destinazione ( $\%ebx - \%edx * 4$ ) in modo tale da retrocedere sull'array di n posizioni pari al valore del registro usato.

`movl temp, %edx`

Viene ripristinato il valore precedentemente contenuto in %edx.

### 8.6.4 Sub memoria, registro

<pre>subl 4(%esp,%ebx,4),%ebx</pre>	<pre>movl %edx, temp movl data_items+512(,%ebx,8), %ebx movl data_items+512(,%ebx,8), %ebx movl 4(%esp, %ebx, 4), %edx movl data_items_negative(,%edx,4), %edx movl data_items+512(%ebx,%edx,4), %ebx movl temp, %edx</pre>
-------------------------------------	---

<code>movl %edx, temp</code>	Viene salvato in temp il contenuto del registro %edx per evitare la perdita del valore salvato.
<code>movl data_items+512(,%ebx,8), %ebx</code>	Utilizzato per andare a recuperare la cella di memoria che punta al valore contenuto dal registro %ebx.
<code>movl 4(%esp, %ebx, 4), %edx</code>	Viene salvato in %edx il valore della cella di memoria puntata da $4 + \%esp + \%ebx * 4$ , per poi utilizzarlo come spostamento sull' array locazione-valore.
<code>movl data_items_negative(,%edx,4), %edx</code>	Viene recuperato il valore negativo della locazione di memoria precedentemente salvata in %edx per effettuare la sottrazione.
<code>movl data_items+512(%ebx,%edx,4), %ebx</code>	Parte fondamentale del paper, implementazione della sottrazione tramite l' uso della mov con accesso alla memoria. Viene puntato sul nostro array il risultato del registro di destinazione $\%ebx - \%edx * 4$ in modo tale da retrocedere sull' array di n posizioni pari al valore del registro usato.
<code>movl temp, %edx</code>	Viene ripristinato il valore precedentemente contenuto in %edx

### 8.6.5 Sub registro, memoria

<pre>subl %ebx,4(%esp,%ebx,4)</pre>	<pre>movl %edx, temp movl %ecx, temp2 movl 4(%esp, %ebx, 4), %edx movl %ebx, %ecx movl data_items_negative(,%ecx,4), %ecx movl data_items+512(,%edx,8), %edx movl data_items+512(,%edx,8), %edx movl data_items+512(%edx,%ecx,4), %edx movl %edx, 4(%esp, %ebx, 4) movl temp, %edx movl temp2, %ecx</pre>
-------------------------------------	---

<code>movl %edx, temp</code>	Viene salvato in temp il contenuto del registro %edx per evitare la perdita del valore salvato.
<code>movl %ecx, temp2</code>	Viene salvato in temp2 il contenuto del registro %ecx per evitare la perdita del valore salvato.
<code>movl 4(%esp, %ebx, 4), %edx</code>	Viene salvato in %edx il valore puntato dalla cella di memoria corrispondente a $4 + \%esp + \%ebx * 4$ , per poi utilizzarlo come spostamento sull'array locazione-valore.
<code>movl %ebx, %ecx</code>	Viene salvato il valore contenuto nel registro %ebx che poi mi servirà per effettuare l'operazione di spostamento sull'array locazione-risultati.
<code>movl data_items_negative(,%ecx,4), %ecx</code>	Viene recuperato il valore negativo del registro precedentemente salvato in %edx per effettuare la sottrazione.
<code>movl data_items+512(,%edx,8), %edx</code>	Utilizzato per andare a recuperare la cella di memoria che punta al valore contenuto dal registro %edx.
<code>movl data_items+512(%edx,%ecx,4), %edx</code>	Parte fondamentale del paper, implementazione della sottrazione tramite l'uso della mov con accesso alla memoria. Andiamo a puntare sul nostro array dei risultati il risultato del registro di destinazione %ebx - $\%edx * 4$ in modo tale da retrocedere sull'array di n posizioni pari al valore del registro usato.
<code>movl %edx, 4(%esp, %ebx, 4)</code>	Viene salvato nella cella di memoria che rappresenta il nostro registro di destinazione il valore ottenuto dalla nostra operazione di sottrazione.
<code>movl temp, %edx</code>	Viene ripristinato il valore precedentemente contenuto in %edx.
<code>movl temp2, %ecx</code>	Viene ripristinato il valore precedentemente contenuto in %ecx.

## Capitolo 9

# Test sull'applicazione

In informatica, il test del software è un procedimento, che fa parte del ciclo di vita del software, utilizzato per individuare le carenze di correttezza, completezza e affidabilità delle componenti software in corso di sviluppo e se quest'ultimo rispetta i requisiti richiesti dall'utente. Consiste nell'eseguire il software da collaudare, da solo o in combinazione ad altro software di servizio, per valutarne il comportamento. I test devono essere rieseguiti dopo ogni cambiamento per controllare che le modifiche apportate per raggiungere un requisito non ne abbiano corrotto delle altre.

Solitamente la fase di test è divisa in 3 parti:

- Test di sviluppo: eseguito dagli stessi sviluppatori
- Test di release: test black box (qualcuno di diverso dagli sviluppatori, non deve conoscere la programmazione software)
- Test degli utenti: utenti nel loro contesto di riutilizzo devono testare il software

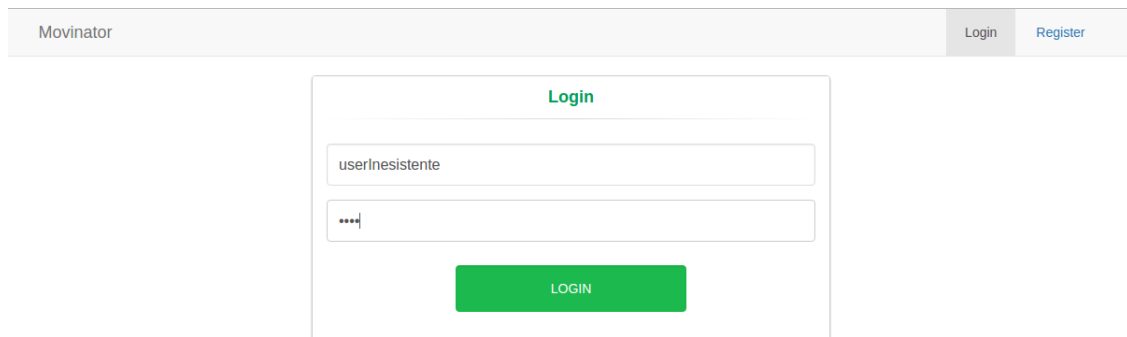
Abbiamo effettuato sia test di sviluppo che test in black box. Di seguito vengono riportati alcuni dei test di sviluppo; per semplicità abbiamo omissso la documentazione relativa ai test effettuati in black box.



## 9.1 Test funzionalità applicazione

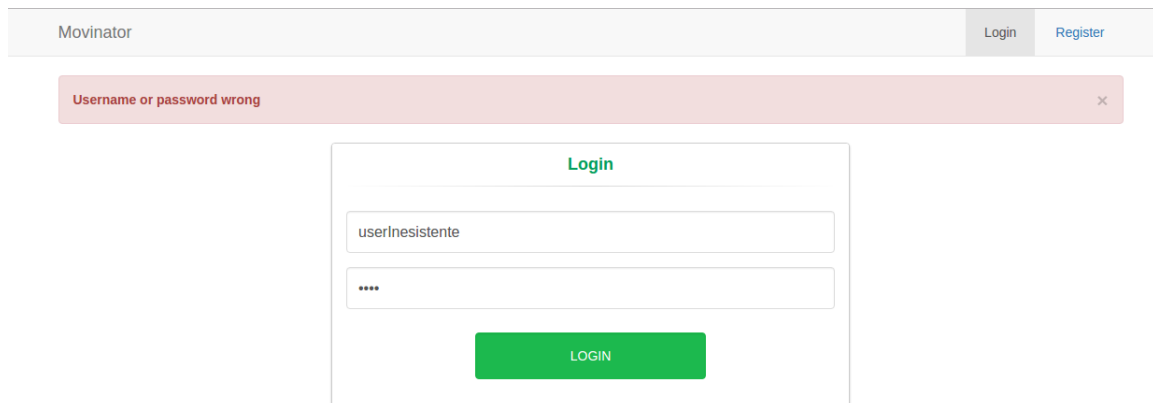
### 9.1.1 Controllo login

L'applicazione non lascia accedere all'area degli utenti autorizzati, gli utenti che non sono correttamente registrati alla base di dati. Qualora un utente cercasse di autenticarsi con delle credenziali errate verrà notificato con un flash message di errore.



The screenshot shows the top navigation bar with the text "Movinator" on the left and "Login" and "Register" links on the right. Below the navigation bar is a "Login" form. The form has a title "Login" in green. It contains two input fields: the first is labeled "userInesistente" and the second is labeled "\*\*\*\*". Below the input fields is a green button labeled "LOGIN".

Figura 9.1: Inserimento di un username e password errati



The screenshot shows the same login form as in Figure 9.1, but with an additional error message. A red flash message box is displayed above the form, containing the text "Username or password wrong" and a close button (X). The form itself remains the same, with the "userInesistente" username and "\*\*\*\*" password, and the "LOGIN" button.

Figura 9.2: Risposta da parte dell'applicazione

### 9.1.2 Controllo registrazione

Per evitare di sporcare la base di dati con la registrazione di utenti incompleti o con migliaia di tuple nulle, abbiamo implementato dei controlli sull'immissione dei campi per la registrazione alla nostra applicazione.

#### Campi nulli

Quando un utente invia la richiesta di registrazione non completando i campi obbligatori verrà notificato cercando gli attributi mancanti in rosso.

The screenshot shows the 'Movinator' application header with 'Login' and 'Register' links. The 'Register' link is active. Below the header is a form titled 'Insert your data'. The form contains several input fields, each with a red border indicating a validation error. The fields and their corresponding error messages are:

- Username:** Username cannot be blank.
- Password:** Password cannot be blank.
- Repat password:** (No error message shown)
- Email:** Email cannot be blank.
- Country:** Country cannot be blank.
- Sex:** Sex cannot be blank.
- Job:** Job cannot be blank.

At the bottom of the form is a green 'REGISTER' button.

Figura 9.3: Risposta da parte dell'applicazione

## Username già presente

Un altro controllo fondamentale è quello di evitare che due utenti si registrino con lo stesso username. Quando un utente prova ad inviare una richiesta di registrazione con un nome già presente nella base di dati verrà notificato da un messaggio di errore.

---

Username already present

Insert your data

user

\*\*\*\*

Repat password

email@gmail.com

American Samoa

Man

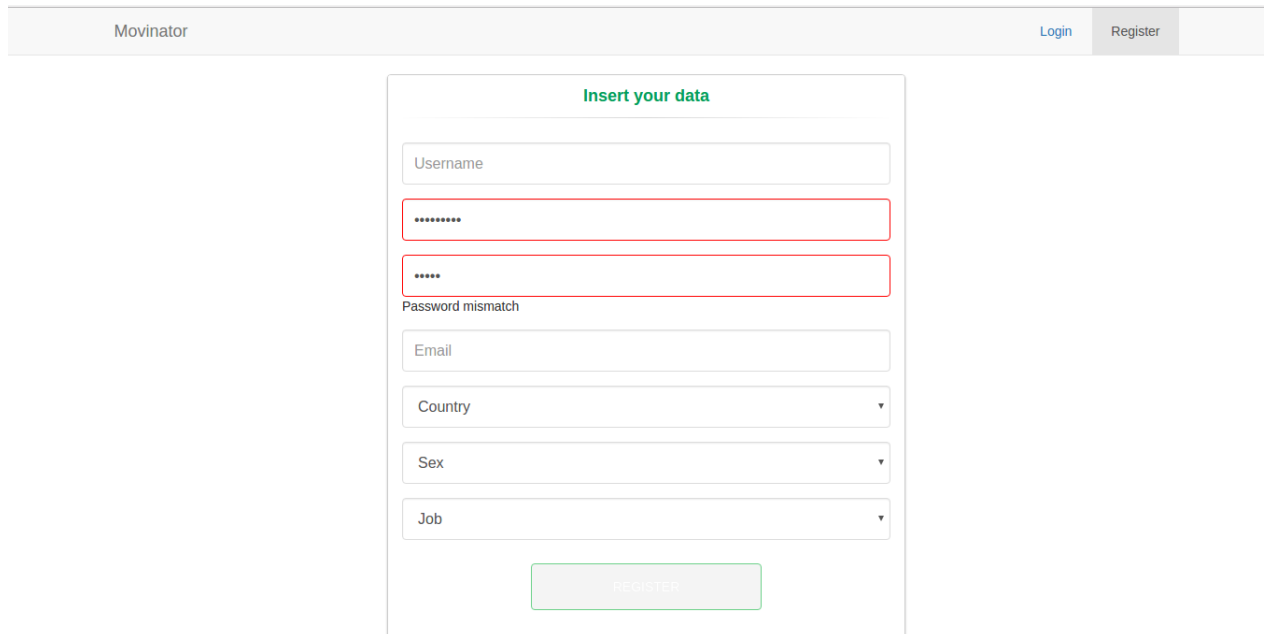
White Hat

REGISTER

Figura 9.4: Risposta da parte dell'applicazione

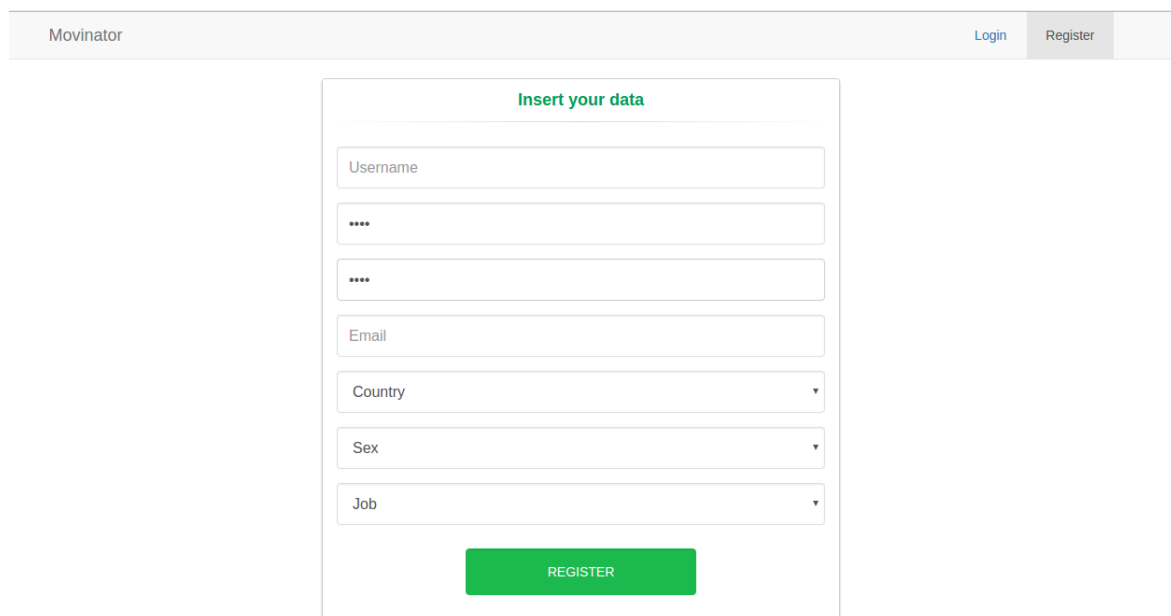
### Password mismatch

Per evitare che eventuali errori di digitazione nell'inserimento della password da parte dell'utente portino alla creazione di dati inconsistenti nella base di dati, abbiamo implementato un ulteriore campo con un controllo sulla password. Se la seconda password inserita non corrisponde con la prima il bottone di invio della richiesta verrà disattivato. Ci siamo serviti dell'utilizzo di uno script jQuery per implementare un controllo real time che disabilita il bottone di login qualora le due password non corrispondessero



The screenshot shows a web application header with 'Movinator' on the left and 'Login' and 'Register' links on the right. The 'Register' link is highlighted. Below the header is a registration form titled 'Insert your data'. The form contains fields for Username, Password (masked with 7 dots), Password (masked with 4 dots), Email, Country (dropdown), Sex (dropdown), and Job (dropdown). A red border highlights the two password fields, and the text 'Password mismatch' is displayed below them. The 'REGISTER' button at the bottom is disabled and has a light gray background.

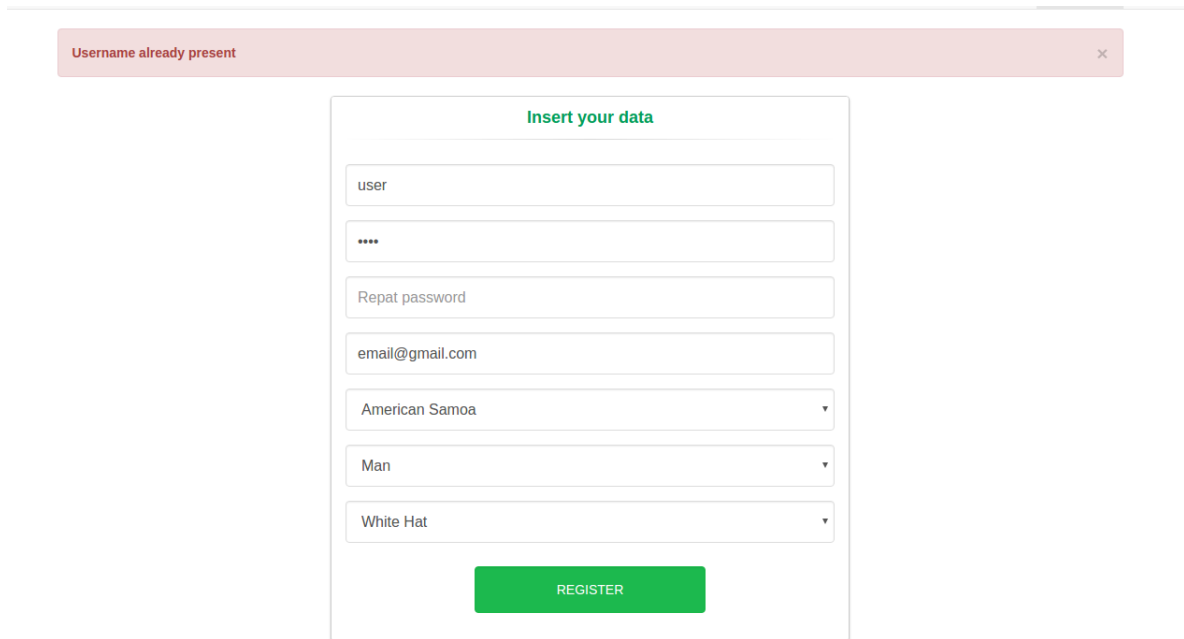
Figura 9.5: Digitazione di una password inconsistente



The screenshot shows the same web application header as Figure 9.5. The registration form titled 'Insert your data' is shown with the Username field containing '\*\*\*\*'. The Password fields are empty. The 'REGISTER' button at the bottom is now active and has a green background.

Figura 9.6: Duplicazione nome utente

**Controllo inserimento utente** Abbiamo inserito un controllo per impedire che si possano registrare due persone con il medesimo username. In caso di inserimento di username già presente nella basi di dati il sistema risponderà come segue.

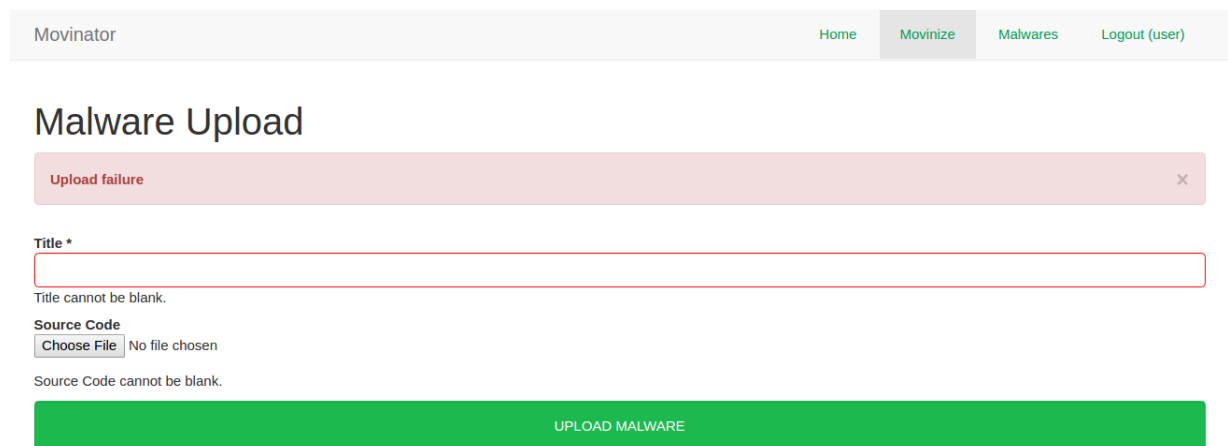


The screenshot shows a web interface for user registration. At the top, a red error banner displays the message "Username already present" with a close button. Below this is a form titled "Insert your data" in green. The form contains several input fields: "user" (containing "user"), a password field (containing "\*\*\*\*"), "Repat password", "email@gmail.com", a country dropdown menu (showing "American Samoa"), a gender dropdown menu (showing "Man"), and a role dropdown menu (showing "White Hat"). At the bottom of the form is a green "REGISTER" button.

Figura 9.7: Duplicazione nome utente

### 9.1.3 Fallimento caricamento malware

Abbiamo effettuato dei controlli sull'esito dell'operazione di caricamento dei malware. Le possibilità di errore vanno dalla non compilazione tutti i campi che compaiono nel form di carimento a problemi di salvataggio su filesystem o sulla base di dati. In questi casi verrà notificato con un messaggio di errore che il caricamento non è andato a buon fine e nel caso in cui non vengano compilati i campi obbligatori apparirà un messaggio di notifica dei campi che non sono stati inseriti.



The screenshot shows a web interface for malware upload. At the top, a navigation bar includes "Movinator", "Home", "Movinize", "Malwares", and "Logout (user)". Below the navigation bar is a section titled "Malware Upload". A red error banner displays the message "Upload failure" with a close button. Below the banner, the form has a "Title \*" field (empty) with a red border and a message "Title cannot be blank." Below this is a "Source Code" section with a "Choose File" button and the text "No file chosen". Below this is a message "Source Code cannot be blank." At the bottom of the form is a green "UPLOAD MALWARE" button.

Figura 9.8: Faillimento upload

## 9.2 Test sicurezza applicazione

Uno dei nostri requisiti iniziali era quello di rendere sicura la nostra applicazione tramite alcuni accorgimenti. Molti accorgimenti di sicurezza sono ereditati dal framework, inoltre utilizziamo un'estensione per la sanitizzazione dei dati passati in GET o POST. Di seguito riportiamo alcuni dei test effettuati da noi.

### Inserimento dati inconsistenti nel database tramite richieste curl

Questo test serve per provare che il metodo `actionCreate` presente nel controllore del modello `User` effettua una valutazione dei dati lato server. Il test viene effettuato su questo metodo perchè è una delle tre azioni che possono essere eseguite da utente non autenticato. Per le altre azioni dove viene richiesta un'autenticazione sarebbero entrati in funzione i controlli effettuati dalle `accessRules` del framework di Yii.

```
tommaso@tommaso-XPS-15-9550:~$ curl 'http://www.movinator.local/movinator/index.php/user/create' -H 'Pragma: no-cache' -H 'Origin: http://www.movinator.local' -H 'Accept-Encoding: gzip, deflate' -H 'Accept-Language: it-IT, it;q=0.8, en-US;q=0.6, en;q=0.4' -H 'Upgrade-Insecure-Requests: 1' -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.106 Safari/537.36' -H 'Content-Type: application/x-www-form-urlencoded' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8' -H 'Cache-Control: no-cache' -H 'Referer: http://www.movinator.local/movinator/index.php/user/create' -H 'Cookie: PHPSESSID=86nsqrjs0uelrrjn7itap7ena7' -H 'Connection: keep-alive' --data 'User%5Busername%5D=userprova&User%5Bpassword%5D=eeee&User%5Bemail%5D=eeee%40gmail.com&User%5Bcountry%5D=CC&User%5Bsex%5D=m&User%5Bjob%5D=black+hat&yt0=Register' --compressed
```

Figura 9.9: Visualizzazione richiesta

```
tommaso@tommaso-XPS-15-9550:~$ curl 'http://www.movinator.local/movinator/index.php/user/create' -H 'Pragma: no-cache' -H 'Origin: http://www.movinator.local' -H 'Accept-Encoding: gzip, deflate' -H 'Accept-Language: it-IT, it;q=0.8, en-US;q=0.6, en;q=0.4' -H 'Upgrade-Insecure-Requests: 1' -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.106 Safari/537.36' -H 'Content-Type: application/x-www-form-urlencoded' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8' -H 'Cache-Control: no-cache' -H 'Referer: http://www.movinator.local/movinator/index.php/user/create' -H 'Cookie: PHPSESSID=86nsqrjs0uelrrjn7itap7ena7' -H 'Connection: keep-alive' --data 'User%5Busername%5D=userprova&User%5Bpassword%5D=eeee&User%5Bemail%5D=eeee%40gmail.com&User%5Bcountry%5D=CC&User%5Bsex%5D=m&User%5Bjob%5D=black+hat&yt0=Register' --compressed
```

Figura 9.10: Visualizzazione risposta

Come si pu ben vedere la richiesta curl non ha ottenuto risposta, quindi la nostra prova ha avuto successo, e il dato non è stato inserito nel database.

+ Opzioni										
			id	username	password		email	country	sex	job
<input type="checkbox"/>	Modifica			1	admin	\$2y\$13\$9FZG3oJWHd4SMBeXTY5GeBpU.8hOMMBjCjX77ARhh...	andreaicolato94@gmail.com	IT	m	white ha
<input type="checkbox"/>	Modifica			2	user	\$2y\$13\$VOHYGgWcMO7Yis2ZGUTue4agDEVIH2QxhDhPYevgmL...	pino@pino.it	IT	m	black ha
<input type="checkbox"/>	Modifica			3	user1	\$2y\$13\$ArW4XIMJmISglRIQyd6uuOrDxDyDjVzMMor.ZIXxM...	user1@gmail.com	AL	m	black ha
<input type="checkbox"/>	Modifica			4	user22	\$2y\$13\$BsuMUh7OUXN0jako1f7O79A9biXD3pyKyZrDzug5V...	pass@gmail.com	AF	m	black ha
	<input type="checkbox"/> Seleziona tutto	Se selezionati:								
										Esporta

Figura 9.11: Visualizzazione Database

## Accesso tramite url path

Un altro possibile "attacco" è quello di provare ad accedere a parti della web application senza essersi correttamente autenticati. Questo problema per la maggior parte delle volte è causato dal programmatore che non ha impostato correttamente le ACL dell'applicazione. In questo caso entrano in funzione le accessRules.



Figura 9.12: URL di visualizzazione di un dato malware



Figura 9.13: Redirect alla homepage

L'applicazione se non autenticati correttamente reindirizza sempre alla pagina di login.

## Capitolo 10

# Design pattern

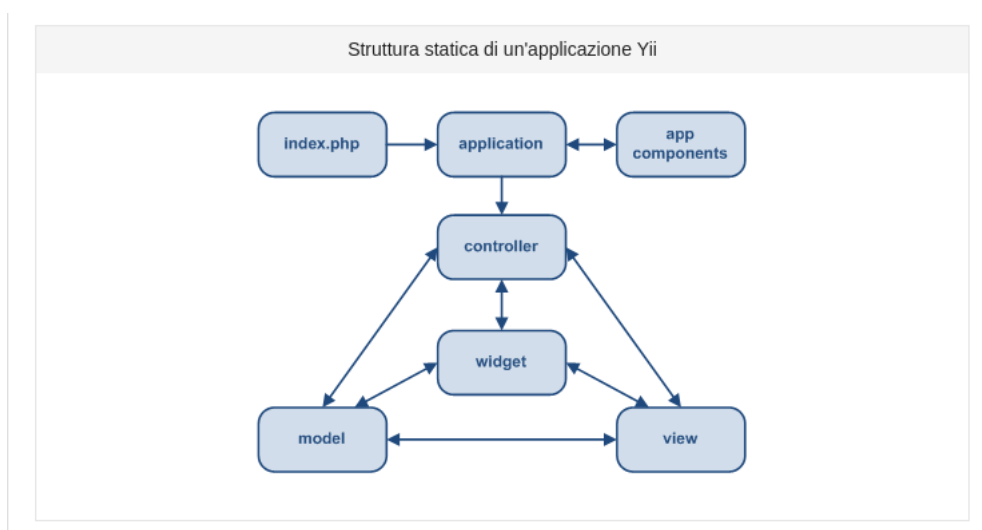
I design pattern (letteralmente “pattern di progettazione”) sono pattern (cioè nati per perseguire un intento: la soluzione di un problema) che utilizzano metodi e classi di un linguaggio OO. Inoltre i design pattern sono ad un livello più elevato rispetto ai linguaggi di programmazione Object Oriented perché indipendenti dagli stessi.

Dato che in fase di progettazione abbiamo scelto di utilizzare il framework Yii, non abbiamo avuto bisogno di implementare alcun design pattern visto che il framework stesso mette a disposizione una suite di strumenti che includono i concetti di alcuni dei principali design pattern. Di seguito vengono descritti alcuni di questi design pattern.

### 10.1 MVC

Yii implementa il design pattern model-view-controller (MVC), che è largamente adottato nella programmazione Web. L'obiettivo di MVC è quello di separare la business logic [logica di funzionamento] dalle considerazioni relative all'interfaccia utente, cosicché gli sviluppatori possono modificare ciascuna parte più facilmente senza influenzare le altre. Nel MVC il model rappresenta le informazioni (i dati) e la business logic; la view contiene elementi dell'interfaccia utente come testi, form di inserimento dati; e il controller gestisce le comunicazioni tra model e view.

Yii, oltre ad implementare MVC, introduce un front-controller, chiamato Application il quale incapsula il contesto di esecuzione per il processo di una richiesta. Application raccoglie alcune informazioni sulla richiesta dell'utente e poi le smista al controller appropriato per ulteriori manipolazioni.





## 10.2 Singleton

Questo design pattern è usato per assicurare che una classe abbia una sola istanza ed un unico punto di accesso globale. In molte situazioni c'è la necessità di garantire l'esistenza di un unico oggetto di una classe

Le classi Singleton vengono progettate con i costruttori privati per evitare la possibilità di istanziare un numero arbitrario di oggetti della stessa. Esiste un metodo statico con la responsabilità di assicurare che nessuna altra istanza venga creata oltre la prima, restituendo contemporaneamente un riferimento all'unica esistente.

La classe mantiene all'interno il riferimento all'unica istanza Singleton della classe.

L'istanza alla prima esecuzione del metodo statico (inizializzazione pigra) oppure contemporaneamente alla definizione della variabile di istanza riferimento all'oggetto. La classe contiene poi tutti i metodi, le proprietà e gli attributi tipici dell'astrazione per cui è stata concepita.

L'applicazione Yii è un singleton, infatti se noi andiamo nel file index.php troviamo lo stato

```
Yii::createWebApplication($configFile)->run();
```

Questo è lo stato con il quale creiamo l'istanza della nostra applicazione Yii è una classe che estende YiiBase. class Yii extends YiiBase createWebApplication() è una funzione statica nella classe YiiBase. Questa funzione ritorna un oggetto CWebApplication class.

```
public static function createWebApplication($config=null)
{
    return self::createApplication('CWebApplication',$config);
}
```

```
createApplication('CWebApplication',$config)
```

creazione dell'oggetto di CWebApplication e ritorno di tale oggetto.

## Capitolo 11

# Gestione evoluzione

Non abbiamo dovuto affrontare questa problematica, perchè essendo un applicazione sviluppata in un mese di lavoro e non avendo clienti a cui far fronte, non siamo incappati in cambi di requisiti in corso d'opera ed evoluzione dei software utilizzati. Abbiamo seguito dall'inizio alla fine dell'implementazione della nostra applicazione le nostre idee di sviluppo.