

---

# Lupus in Tabula

---

Edoardo Morassutto

Ultimo aggiornamento  
26 aprile 2014

Tutto il sorgente è disponibile all'indirizzo: <https://github.com/lupus-dev/lupus>

Per contattare l'amministratore: [edoardo.morassutto@gmail.com](mailto:edoardo.morassutto@gmail.com)

Puoi contattarmi su Facebook se mi hai tra gli amici!

Se vuoi dare una mano ogni tanto forka il repository `/lupus-dev/lupus` su GitHub e inviaci una **Pull Request**, se sei stato bravo la mergiamo volentieri!

Se vuoi diventare uno sviluppatore chiedimelo e vedrò di accontentarti!

## 1 Introduzione

Il progetto viene scritto in PHP. Ovviamente non può mancare l'HTML, CSS, JavaScript, database, git, SASS, ed altro...

### 1.1 Pacchetti software

Il progetto verrà avviato su una macchina con questi pacchetti (e dipendenze):

- Apache2 (qualunque versione recente dovrebbe andare)
  - `Mod_rewrite` (necessaria per il rewrite dell'url)
- PHP 5.3 (compatibile con i più comuni webserver gratuiti)
  - estensioni varie da definire (le solite comunque)
- MySQL (compatibile con i più comuni webserver gratuiti)
- SASS(solo lato compilazione del CSS)
  - Quindi Ruby...

## 1.2 Motivazione delle scelte

### Apache2

Ormai questo è un must dei webserver, nulla da ridire, facile da configurare, veloce, compatibile con tutti i servizi di webhosting gratuiti.

È necessario abilitare l'estensione `mod_rewrite` (solitamente è già attiva) per consentire il rewrite degli indirizzi e rendere più accattivante il servizio del gioco.

Tramite il file `.htaccess` vengono configurate l'estensioni usate, facile il versioning e la gestione della configurazione.

### PHP

Dato che il server del gioco verrà scritto in PHP, è necessario che PHP sia installato e configurato correttamente.

Il progetto verrà scritto in PHP 5.3 (la versione attuale di Altervista) per permettere una facile migrazione di host nel caso sia necessario e per renderlo il più retrocompatibile possibile. Alcune funzioni sono state marcate come *deprecated* in 5.4, sapendolo cerchiamo di evitarle... Ad esempio non useremo l'estensione `mysql` ma la nuova `mysqli` con approccio ad *oggetti*. È necessario quindi che sia abilitata se non lo fosse già.

### MySQL

Per quanto mi possa piacere di più PostgreSQL sono costretto a scegliere di usare MySQL per le stesse ragioni per cui useremo PHP 5.3. PostgreSQL non è supportato da Altervista e da altri webhosting gratuiti.

La versione è abbastanza ininfluente, non useremo funzioni particolarmente recenti.

### SASS

SASS è un preprocessore di CSS, permette di scrivere il CSS molto più semplicemente e di applicare delle particolari formattazioni al file `.css` in maniera del tutto automatica. Le più importanti differenze tra CSS e SASS sono:

- In SASS si possono dichiarare le variabili ed effettuare operazioni con esse
- Si può decidere di comprimere il file `.css` di output
- Si possono annidare le dichiarazioni (vedi manuale del SASS)
- È molto più facile da leggere e da mantenere
- Si possono includere altri fogli SASS dentro un file SASS (aka *import*)

SASS richiede che sia installato anche Ruby per funzionare...

Usando NetBeans è possibile abilitare la compilazione automatica e trasparente dei file `.scss` ad ogni salvataggio.

## 2 Il gioco

Lupus in Tabula è un *gioco di ruolo* molto complesso. Per imparare a giocare si può fare riferimento al nostro amato amico Google...

Questa è una semplice guida per iniziare a giocare:

[http://www.davincigames.net/lit/lupus\\_in\\_tabula\\_4th\\_regole.pdf](http://www.davincigames.net/lit/lupus_in_tabula_4th_regole.pdf)

Ignorare le carte e pensare in grande per scrivere un server web...

### 2.1 Struttura dei componenti

Ovviamente perché si possa chiamare *gioco* è necessario che tutto abbia una struttura solida e chiara. Vengono quindi spiegate le principali entità del gioco. Tutti i riferimenti ai *nomi brevi* sono chiariti nella sezione apposita.

#### 2.1.1 Gli utenti

- Ogni utente che si registra ha un identificativo univoco che fa da chiave primaria
- Ogni utente ha uno **username** univoco che usa per identificarsi e per accedere al sito
- Lo **username** è un *nome breve*
- Ogni utente ha un *livello* pubblico che determina i privilegi a cui l'utente può accedere
- Ogni utente ha una email che viene usata per verificare l'account e notificare qualcosa... (*da definire*)

#### 2.1.2 Le stanze

- Ogni utente può creare un certo numero di stanze (in base al livello)
- Ogni stanza può contenere più partite, una sola però può essere attiva
- Ogni stanza ha un nome breve **room\_name**
- Ogni stanza ha una descrizione con lo stesso grado di accessibilità della stanza
- Le stanze possono essere private o pubbliche
- Ogni utente può visualizzare una stanza pubblica
- Solo gli utenti partecipanti e/o autorizzati possono visualizzare una stanza privata
- Ogni stanza ha un amministratore, colui che ha creato la stanza
- Le stanze possono essere eliminate

### 2.1.3 Le partite

- Ogni partita è interna ad una stanza
- Ogni partita ha un nome breve unico per quella stanza (`game_name`)
- Ogni partita ha una descrizione scelta dall'amministratore della stanza
- L'amministratore della partita è l'amministratore della stanza corrispondente
- Possono essere create delle partite solo se non ci sono partite attive
- Gli utenti possono essere invitati nella partita oppure possono richiedere all'amministratore di farne parte
- La partita può iniziare solo se è presente il numero di giocatori richiesto
- La partita viene avviata dall'amministratore
- L'amministratore può espellere un giocatore
- L'amministratore può terminare la partita
- I ruoli dei giocatori vengono rilevati solo al termine della partita
- La chat della partita rimane visibile solo ai giocatori
- Eventuali altre chat vengono disabilitate alla fine della partita

### 2.1.4 I livelli

- Ogni utente ha un livello
- Il livello è solo crescente (non si può essere degradati tranne dal `Game Master`)
- Un utente bannato ha livello uguale a *zero* (da definire...)

### 2.1.5 Nomi brevi

Tutti i riferimenti ai nomi brevi sono da intendere che rispettano le seguenti caratteristiche:

- Hanno una lunghezza di al più 10 caratteri
- Hanno una lunghezza di almeno un carattere
- Sono formati da soli caratteri ASCII alfanumerici
- Il primo carattere è una lettera
- Ogni nome breve è unico rispetto al proprio contesto

## 2.2 Struttura del server

### Modulare

Il *server* e le **API** sono gestite nel modo più modulare possibile:

Deve essere possibile avere una cartella dove mettere dei file **PHP** contenenti i ruoli. Aggiungere e togliere dei ruoli deve essere una semplice modifica del relativo file in questa cartella.

Non deve essere presente un file dove vengono registrati i ruoli, deve essere gestito in maniera automatica e dinamica.

### Orientato agli oggetti

Ogni ruolo deve essere un oggetto che deriva dalla classe **Role**.

I file PHP contenenti le informazioni di un ruolo devono chiamarsi **role.Ruolo.php** dove ruolo è il nome della classe e del ruolo con l'iniziale *maiuscola*.

Il nome del ruolo è un nome breve.

Se un giocatore ha un ruolo non esistente o non abilitato la partita termina a causa di un errore interno.

## 2.3 Ruoli

I ruoli da implementare fin da subito potrebbero essere: (il colore indica la fazione, la stella il mana)

- **Lupo★** Durante la notte i *lupi* votano chi eliminare, se almeno il 50%+1 dei *lupi* vivi votano la stessa persona, questa è una candidata a morire;
- **Guardia★** Durante la notte la *guardia* può scegliere di proteggere una persona, se i *lupi* quella notte decidessero di ucciderla, essa non muore;
- **Medium★** Il *medium* durante la notte può scegliere di guardare un giocatore morto, lui saprà se quel giocatore aveva un mana buono o cattivo;
- **Veggente★** Il *veggente* può fare le stesse operazioni del *medium*, solo sui giocatori vivi;
- **Paparazzo★** Il *paparazzo* durante la notte sceglie una persona da pedinare, vengono riportati sul giornale della mattina seguente tutti i giocatori che hanno visitato il personaggio *paparazzato*;
- **Criceto mannaro★** É un giocatore normale, senza poteri speciali. Se la partita termina e lui è ancora vivo allora vince solo lui e non la sua fazione;
- **Assassino★** L'*assassino* una sola volta nella partita può scegliere una persona e ucciderla;
- **Massone★** I *massoni* non hanno poteri però hanno una chat dedicata e quindi si conoscono tra loro;
- **Contadino★** I *contadini* non hanno poteri...

- **Pastore★** I *pastori* possono scegliere di sacrificare delle pecore per salvare dei giocatori dalle grinfie dei lupi;
- **Falso★** Il *falso* può scegliere tra una serie di proposizioni false e renderle pubbliche sul giornale;
- **Prescelto★** Il *prescelto* all'inizio della partita conosce 4 proposizioni, 2 vere e 2 false;
- **Sindaco★** Il *sindaco* è un contadino che non può essere messo al rogo nella votazione diurna;

## 2.4 Stati partita

Ogni partita contiene il campo *status*, un valore intero che rappresenta lo stato di gioco della partita.

Lo stato della partita divide ogni partita in 3 fasi:

### 2.4.1 Fase pre-partita

Questa fase è presente solo prima dell'avvio della partita e serve per salvare ed aggiustare i dettagli della partita, come il numero di giocatori, di ruoli, la descrizione, ecc. . .

Questa fase è identificata da codici nell'intervallo:

$$0 \leq x < 100$$

I codici riconosciuti sono:

- **0: Setup** Impostazione della partita

### 2.4.2 Fase in-partita

Questa fase è presente poco prima dell'inizio della partita e durante tutto il corso della partita.

Viene identificata da codici nell'intervallo:

$$100 \leq x < 200$$

I codici riconosciuti sono:

- **100: NotStarted** La partita è attiva e i giocatori possono iniziare ad entrare
- **101: Started** La partita è in corso e i giocatori non possono più entrare

### 2.4.3 Fase terminata in modo corretto

Questa fase si verifica quando la partita termina in modo corretto e viene scelta una squadra vincitrice.

I codici che la identificano sono compresi nell'intervallo:

$$200 \leq x < 300$$

I codici riconosciuti sono:

- **200 + y: Win $y$**  La partita è terminata e ha vinto la squadra  $y$  ( $0 \leq y < 99$ )
- **299: DeadWin** La partita è terminata perché tutti i giocatori sono morti

### 2.4.4 Fase terminata in modo inaspettato

Questa fase si verifica quando la partita viene interrotta prima della sua normale fine. In questo caso non viene designato alcun vincitore.

I codici che identificano questa fase sono compresi dall'intervallo:

$$300 \leq x < 400$$

I codici riconosciuti sono:

- **300: TermByAdmin** L'amministratore della stanza ha fatto terminare la partita
- **301: TermBySolitude** Un numero eccessivo di giocatori hanno abbandonato la partita
- **302: TermByVote** È stato raggiunto il *quorum* per terminare la partita
- **303: TermByBug** Un errore interno del server ha fatto terminare la partita per preservare l'integrità del server
- **304: TermByGameMaster** Il *GameMaster* ha deciso di terminare la partita

## 2.5 Misura del tempo

Il tempo in Lupus in Tabula si suddivide in tre *tipi*:

- Arrivo al villaggio
- Giorno
- Notte

Il passare del tempo viene memorizzato nel campo `day` di una partita e il suo *tipo* viene distinto usando la seguente formula:

$$day \left\{ \begin{array}{l} \text{se } day = 0 \Rightarrow \text{arrivo al villaggio} \\ \text{se } day \text{ è } \textit{pari} \Rightarrow \text{giorno } \frac{day}{2} + 1 \\ \text{se } day \text{ è } \textit{dispari} \Rightarrow \text{notte } \frac{day}{2} + 1 \end{array} \right.$$

### 3 Il server

Il server è suddiviso in più parti:

- Il *database*
- Le API
- La *web-app*

#### 3.1 Il database

Il *database* è formato da una serie di tabelle spiegate nel dettaglio nella sezione dedicata più avanti.

#### 3.2 Le API

A causa della natura dinamica del gioco sono necessarie delle funzioni accessibili facilmente tramite **JavaScript**. Tramite queste funzioni è possibile far proseguire la partita, ottenere informazioni ed effettuare richieste al server.

#### Funzioni disponibili

Tutte le seguenti funzioni si intendono già nel percorso delle API (es: `/api`).

<sup>α</sup> indica che la funzione non è completa.

- `/login` Effettua il login tramite **username/password** e salva nella *sessione* l'identificativo dell'utente. Devono essere specificati i parametri **username** e **password** tramite **GET**
- `/login/(username)` Sostituisce il metodo precedente. Effettua il login dell'utente specificato. La password va specificata in **GET** come per `/login`. Se viene specificato anche lo **username** in **GET** viene ignorato.
- `/logout` Se l'utente è connesso lo disconnette cancellando la sessione
- `/status`<sup>α</sup> Mostra delle informazioni sul server, come la versione, ecc..
- `/user/(username)` Mostra le informazioni dell'utente specificato
- `/me` Scorciatoia per `/user/(username)` con lo **username** dell'utente
- `/room/(room_name)` Mostra le informazioni della stanza specificata
- `/game/(room_name)/(game_name)` Mostra le informazioni della partita specificata
- `/new_room/(room_name)` Crea una nuova stanza appartenente all'utente. Deve venire specificato il parametro **descr** in **GET**. Può essere specificato il parametro **private** per rendere la stanza privata



- `/new_game/(room_name)/(game_name)` Crea una nuova partita nella stanza specificata. Devono venire specificati i parametri `descr` e `num_players` in GET.

### 3.3 La web-app

L'interfaccia grafica del gioco sarà sviluppata con la libreria **Bootstrap**, la quale rende tutto molto portabile su dispositivi con schermo ridotto.

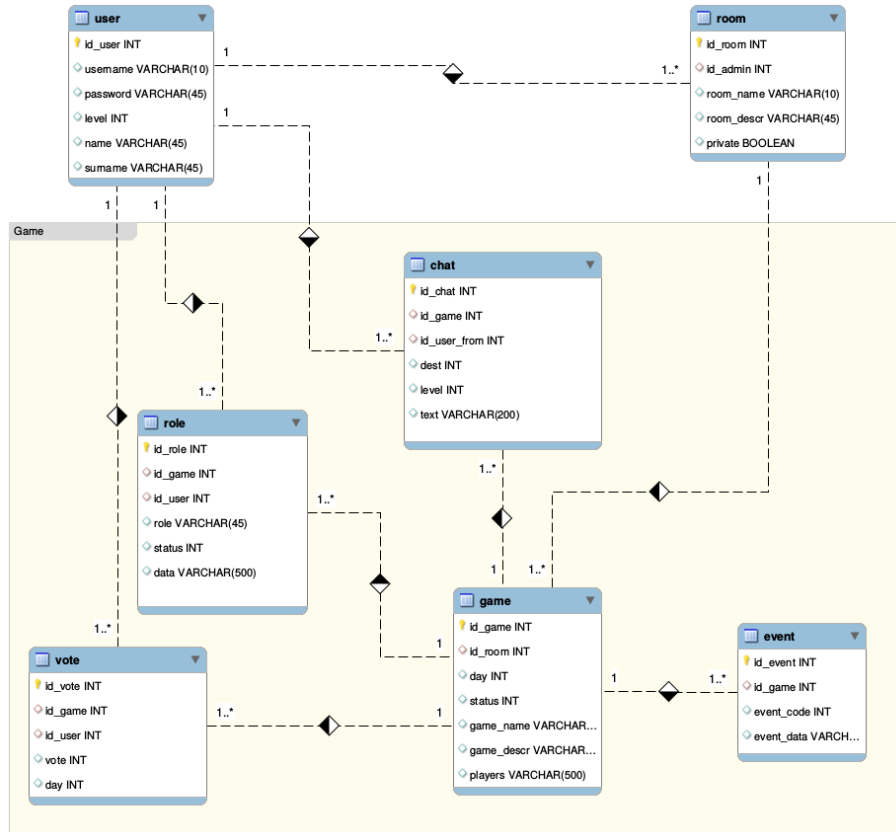
Non ci sarà troppo PHP in queste pagine, il grosso del lavoro sarà il **JavaScript** a farlo, in coppia con le **API**.

#### Pagine

Quello che segue è un elenco delle pagine che devono essere implementate

- `/index` è la pagina che contiene i link alle varie pagine del gioco. Accessibile solo dopo il *login*.
- `/login` è la pagina per accedere al sito
- `/admin` è una pagina che contiene la lista dei collegamenti alle altre pagine di amministrazione
- `/admin/(room_name)` è la pagina di amministrazione di una stanza
- `/admin/(room_name)/(game_name)` è la pagina di amministrazione di una partita
- `/game` è una pagina che contiene la lista dei collegamenti alle altre pagine di gioco
- `/game/(room_name)` è una pagina che contiene le partite di una stanza
- `/game/(room_name)/(game_name)` è la pagina di gioco di una partita
- `/user` è la pagina dell'utente
- `/user/(username)` è la pagina di un utente specifico
- `/status` contiene lo stato del server (numero di partite, di stanze, ecc...)
- `/join` è la pagina per trovare una partita
- `/singup` è la pagina per registrarsi

## 4 Il database



Questa è una *bozza* della struttura del database, ci saranno sicuramente molte modifiche. Lo schema E-R è stato disegnato con il programma **MySQL Workbench**.

### 4.1 Tabella user

In questa tabella vengono memorizzati gli utenti registrati. Ogni utente ha un **id.user** che rimane nascosto al pubblico e identifica all'interno del database uno ed un solo utente. Ogni utente è identificato all'esterno con un **username** unico che rispetta il formato *nome breve*.

La password dell'utente è salvata codificata in **SHA-1**. In futuro potrebbe essere aggiunto un ulteriore livello di protezione aggiungendo del *salt*.

Il livello dell'utente è salvato nella righe dell'utente. Ogni volta che si verifica un evento che potrebbe modificare il livello, si ricalcola il livello e lo si aggiorna solo se è strettamente maggiore di quello precedente.

Ogni utente ha anche un campo **name** e uno campo **surname**

### 4.2 Tabella room

La tabella **room** contiene le informazioni delle stanze.

Ogni stanza ha un identificativo univoco **id.room**, nascosto al pubblico che identifica la stanza all'interno del database.

Ogni stanza contiene l'identificativo dell'utente amministratore della stanza **id\_admin**. Al pubblico la stanza è identificata con un nome **room\_name** il quale è unico. Ogni stanza ha anche una descrizione **room\_descr** che rappresenta il titolo della stanza. Il campo **private** indica se la stanza è privata.

### 4.3 Tabella game

La tabella game raccoglie le informazioni di ogni partita. Ogni partita è identificata all'interno del database con un identificativo unico **id\_game**. Ogni partita è contenuta all'interno di una stanza **id\_room**. Ogni partita è identificata all'esterno con un *nome breve* unico all'interno della stessa stanza **game\_name**, inoltre ogni partita ha una descrizione che rappresenta il titolo della partita **game\_descr**. I campi **day** e **status** sono informazioni utili della partita, l'istante temporale del gioco e lo stato della partita. Il campo **players** contiene una stringa JSON che codifica un vettore degli **username** degli utenti registrati alla partita.

### 4.4 Tabella role

In questa tabella sono memorizzati i ruoli dei giocatori interni ad una partita. Ogni riga è identificata da un campo **id\_role** non pubblico. Ogni ruolo è interno alla partita **id\_game** e relativo all'utente **id\_user**. Il ruolo dell'utente è memorizzato nella stringa **role** la quale identifica un ruolo nella cartella dei ruoli. Il campo **status** indica lo stato dell'utente (vivo o morto). Alcuni ruoli potrebbero richiedere di memorizzare delle informazioni aggiuntive, il campo **data** contiene dei dati salvati da un ruolo, codificati in JSON.

### 4.5 Tabella vote

In questa tabella vengono memorizzati i voti degli utenti. Ogni voto è identificato tramite il campo **id\_vote**, il quale rimane nascosto all'esterno del database. Ogni voto è specifico della partita **id\_game** nel momento **day** e appartiene all'utente **id\_user**. Il suo voto è **vote** il quale può essere l'identificativo di un utente, un voto *nullo* o altro.

### 4.6 Tabella chat

In questa tabella vengono memorizzati i messaggi delle varie chat. Ogni messaggio è identificato all'interno del database da **id\_chat**. Ogni messaggio si riferisce alla partita **id\_game**. Il mittente del messaggio è **id\_user\_from** cioè l'identificativo dell'utente che ha inviato il messaggio. Il destinatario **dest** è un numero intero che può rappresentare vari identificativi, in base a **level** il destinatario può essere un utente, il pubblico, una chat privata, ecc. . .

Il testo del messaggio è `text` il quale non può essere più lungo di 200 caratteri e viene eseguito l'*escape* sia per il database che per il `JavaScript`.

## 4.7 Tabella event

In questa tabella vengono memorizzati gli eventi delle partite.

Ogni evento è identificato dal campo `id_event` e appartiene alla partita `id_game`.

Il tipo di evento che si è verificato è memorizzato in `event_code` e le sue informazioni sono salvate in `event_data`.

### Codici di evento riconosciuti

- **0: GameStart** La partita è appena iniziata  
Parametri nel campo `event_data`:
  - `players` Vettore con gli `username` dei giocatori nella partita
  - `start` Timestamp dell'ora dell'inizio della partita
- **1: Death** Un giocatore è stato ucciso o è stato trovato morto  
Parametri nel campo `event_data`:
  - `dead` Username del giocatore morto
  - `cause` Causa della morte
  - `actor` Causante della morte (killer)

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Pacchetti software . . . . .	1
1.2	Motivazione delle scelte . . . . .	2
<b>2</b>	<b>Il gioco</b>	<b>3</b>
2.1	Struttura dei componenti . . . . .	3
2.1.1	Gli utenti . . . . .	3
2.1.2	Le stanze . . . . .	3
2.1.3	Le partite . . . . .	4
2.1.4	I livelli . . . . .	4
2.1.5	Nomi brevi . . . . .	4
2.2	Struttura del server . . . . .	5
2.3	Ruoli . . . . .	5
2.4	Stati partita . . . . .	6
2.4.1	Fase pre-partita . . . . .	6
2.4.2	Fase in-partita . . . . .	6
2.4.3	Fase terminata in modo corretto . . . . .	7
2.4.4	Fase terminata in modo inaspettato . . . . .	7
2.5	Misura del tempo . . . . .	7
<b>3</b>	<b>Il server</b>	<b>8</b>
3.1	Il database . . . . .	8
3.2	Le API . . . . .	8
3.3	La web-app . . . . .	9
<b>4</b>	<b>Il database</b>	<b>10</b>
4.1	Tabella user . . . . .	10
4.2	Tabella room . . . . .	10
4.3	Tabella game . . . . .	11
4.4	Tabella role . . . . .	11
4.5	Tabella vote . . . . .	11
4.6	Tabella chat . . . . .	11
4.7	Tabella event . . . . .	12