
Lupus in Tabula

Edoardo Morassutto

Ultimo aggiornamento
24 aprile 2014

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

Per contattare l'amministratore: 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, JavaScript, CSS, 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 influente, 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

Ignorare a prescindere la roba delle 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 di livello appropriato 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 partita
- 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 un titolo scelto dall'amministratore
- 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 minimo di giocatori
- La partita viene avviata dall'amministratore
- L'amministratore può espellere un giocatore
- L'amministratore può terminare la partita
- Lo stato della partita e i ruoli dei giocatori vengono rilevati solo al termine della partita
- La chat della partita rimane visibile solo ai giocatori
- Eventuali altre chat vengono disabilite 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 dall' `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
- Sono formati da soli caratteri ASCII alfanumerici minuscoli/cifre
- Il primo carattere è una lettera
- Ogni nome breve è unico rispetto al proprio contesto

2.2 Struttura del server

Modulare

Il server (forse 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 `Ruolo`

Il server include tutti i ruoli presenti nella cartella dedicata. Ognuno di questi file contiene una classe e un'istruzione del genere:

```
$ruoli[] = "nomeClasse";
```

Dove `$ruoli` è un vettore che conterrà i nomi delle classi da interpretare come ruoli. Se uno dei file in quella cartella non dovesse contenere quell'istruzione allora quel ruolo non sarà abilitato.

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 tutti i giocatori che hanno visitato il 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

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 (a volte 4):

2.4.1 Fase pre-partita

Questa fase è presente solo prima dell'avvio della partita e server 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

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: attiva ma non iniziata. La partita è attiva e i giocatori possono iniziare ad entrare
- 101: in corso. 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:

- $200 + y$: la partita è terminata e ha vinto la squadra y

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: terminata dall'admin. L'amministratore della stanza ha fatto terminare la partita
- 301: numero di giocatori insufficiente. Un numero eccessivo di giocatori hanno abbandonato la partita
- 302: terminata per votazione. È stato raggiunto il *quorum* per terminare la partita
- 303: terminata per bug. Un errore interno del server ha fatto terminare la partita per preservare l'integrità del server
- 304: terminata dal `GameMaster`. Il `GameMaster` ha deciso di terminare la partita

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`).

^α 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`^α 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 `roles` in `GET`

3.3 La web-app

L'interfaccia grafica del gioco sarà sviluppata con la libreria Bootstrap, la quale rende tutto molto portatile 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.

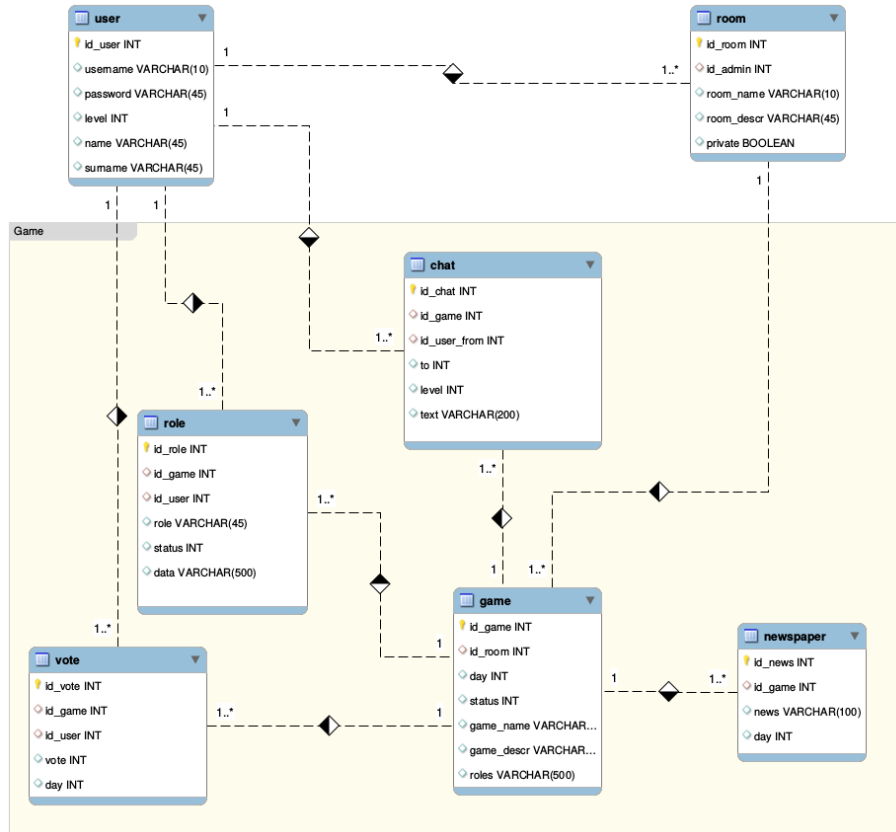
Pagine

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)/(num|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)/(num|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-256. 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.

4.2 Tabella room

La tabella `room` contiene le informazioni di una stanza. 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 `open` nelle prime versioni sarà ignorato perché vanno definiti ancora i permessi per la creazione di stanze e di partite.

4.3 Tabella game

La tabella game raccoglie le informazioni di una singola 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 (in corso, terminata, interrotta, ecc...)

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, morto, ecc...). 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 ma anche 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 `to` 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 newspaper

In questa tabella viene memorizzato il giornale dei vari giorni della partita. Ogni articolo è identificato da `id_news` ed appartiene alla partita `id_game` al tempo `day`. Il testo dell'articolo è memorizzato in `news`.

Indice

1	Introduzione	1
1.1	Pacchetti software	1
1.2	Motivazione delle scelte	2
2	Il gioco	3
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
3	Il server	7
3.1	Il database	7
3.2	Le API	8
3.3	La web-app	8
4	Il database	10
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 newspaper	11