UNIVERSITÀ DEGLI STUDI DI PERUGIA
Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

Tesi di Laurea

STUDIO E SVILUPPO DI UN CONFIGURATORE DI MAZZI DEL GIOCO DI CARTE COLLEZIONABILI MAGIC: THE GATHERING

Laureando
Filippo Mastrini

Relatore
Stefano Bistarelli

ANNO ACCADEMICO
2015-2016

I miei ringraziamenti vanno:

Al ***Professor Stefano Bistarelli***: per il ruolo di committente/relatore.

Al ***Professor Fabio Rossi***: per aver riesumato il demone PHP che incautamente ho ucciso così tante volte, e non solo nel server di mia competenza.

Al ***Professor Andrea Formisano***: per aver acceso insieme la "scintilla" dell'ottimizzazione della query, e la gentilezza portata in quella serata.

A ***Francesco "Galt" Faloci***: per i costanti consigli non solo su PHP, e l'enorme pazienza portata.

A ***Mirco Tracolli***: per gli strumenti e i consigli in ambiente Python.

Ad ***Andrea Nenci***: per i preziosi consigli su HTML passatomi nel breve tempo in cui ho potuto conoscerlo.

Ad ***Alessio Santoru***: per aver condiviso la soluzione di controllo scripts Python con me.

A ***Matteo*** per aver creduto in me, e per il disegno dedicatomi.

Ai miei genitori: per tutto il sostegno datomi in questi lunghi anni.

Grazie veramente di cuore a tutti voi.

Sommario

1) Premessa

Ricordo di un vecchio discorso para-filosofico intrattenuto da compagni di studi, in cui si dichiarava "l'uomo non è divino, perché non è in grado di creare dal completo nulla – tutto ciò che può fare è generare qualcosa dal suo stesso corpo, oppure plasmare ciò che la natura e la realtà gli offre". La mia osservazione, in scatto di megalomania, al tempo, fu "In tal caso, io genero linee di codice che creano una nuova realtà, corretta nella sua singolarità. Allora sono divino? E lo sono anche poeti e musicisti, o gli artisti in generale?"

Quando scelsi di frequentare un'università di informatica, stavo facendo ben più che semplicemente seguire un retaggio tramandato dai miei genitori. Avevo compreso che il continuo sviluppo tecnologico-informatico che si svolge sotto i nostri stessi occhi, ed anche accelerato dalle varie politiche non solo europee, sta lentamente iniziando a mutare le abitudini e di conseguenza le richieste dell'umanità, trascendendo dalla materialità che ci ha interessato per millenni a una nuova realtà "virtuale" – come un tentativo di ribellione, un atavico desiderio di realizzare un tabula rasa e poter sentirsi pertanto padroni di successi e fallimenti, smettendo di essere vittime di regole sin troppo grandi e complesse per essere comprese, a pieno, da una singola mente umana. Il ruolo dell'artigiano, lentamente, partito dal creatore di squisite manifatture si sta lentamente relegando al ruolo di committente per le figure di programmatori e informatici affinché le proprie opere possano essere godute anche in ambienti virtuali – ove possibile. Come poi gli anni mi diedero a dimostrare, per una volta avevo fatto la scelta giusta, dato l'aumento di impieghi riservati alle figure professionali che questa università genera.

La smania di utilizzare la comodità di un mondo virtuale, grande quanto il mondo ma dove qualsiasi destinazione è dietro l'angolo (o meglio, un link), è arrivata nel tempo a modificare molti dei comportamenti dell'uomo. Ciò che un tempo era un tomo pesante ora è diventato un comodo file di qualche megabyte di peso al più, il concetto di "lavoro telematico" ingolosisce sempre di più i lavoratori, che possono finalmente iniziare a lavorare senza lo stress dovuto dal traffico o di un ambiente di lavoro altrimenti insopportabile, intere foreste ci ringraziano (un po' meno i postini) per l'uso di posta elettronica invece della cartacea, più rapida ed efficace, i software di simulazione continuano a permettere sempre più agli uomini di poter testare le loro idee senza recar disturbo ai propri simili, o al mondo su cui vivono, in ambienti sempre più vasti, fino addirittura a quelli più intimi e su cui, forse qualche tempo fa, qualcuno poteva dubitare un'applicazione non-fisica.

Lascio la disquisizione di questa evoluzione – o forse, direzione? - presa dall'uomo a menti più competenti.

Di questi ambienti di simulazione, ne prenderò uno in particolare. Lo svago on-line, basandosi su qualcosa di reale... di cartaceo, nello specifico. Una simulazione che permette ad Alice e a Bob di poter fare qualcosa che la distanza avrebbe reso difficoltoso se non impossibile da realizzare.
La possibilità di condurre un gioco di carte.
Un gioco di carte di ***Magic: The Gathering.***

1.1) Configuratori di Prodotti – Cosa sono

Software house, o programmatori in generale, hanno un lavoro ben determinato: sviluppare software che rispondano a determinate

esigenze, fissate da loro stessi o un committente. Resta comunque il concetto basilare:

- Immaginare una funzione, ovvero un'applicazione, nella realtà
- Chiedersi se è possibile implementarla in un ambiente virtuale, dopo essersi accertati che la cosa può essere realmente fattibile
- Se sì:
  - Pensare a **un** modello (non esiste "*il*" singolo algoritmo totalmente e completamente giusto) di implementazione dell'applicazione o funzione, eventualmente aggiustandosi con delle esigenze "dettaglio" aggiuntive (vuoi per dimenticanza del committente, vuoi perché sinceramente impreviste prima, vuoi per compromesso tra le parti…)
  - Scrivere il codice che di fatto la implementa

È un processo che volendo è applicabile anche nella realtà. Si supponga di avere una piccola collezione di libri. Si vuole ordinarli secondo qualche logica:

- Immaginare la funzione: ordinamento (per ordine alfabetico? Per autore? Per casa editrice?). Viene scelto: ordine alfabetico.
- Chiedersi: è possibile fisicamente ordinare una collezione di libri per ordine alfabetico?
  - Ovviamente sì
- Chiedersi: è possibile, una volta caricati tali libri in un formato virtuale, ordinarli per ordine alfabetico, in tale ambiente virtuale?
  - Sì. E come informatici sappiamo che esiste anche più di un modo per farlo
- Pertanto: pensare a un modello per ordinare.

- Si noti come nella realtà si prendono tutti i libri, uno per uno, e poi si piazzano da sinistra a destra, usando come riferimento il libro "prima" per decretare se il libro osservato dovrebbe andare prima di esso, oppure si trova nella posizione esatta.
- Si decide a questo punto di imporsi una esigenza: un *modello che non sia troppo lungo da implementare*
  - Poiché non è preteso pertanto che l'algoritmo sia efficiente, è sufficiente un modello stile bubble-sort

➢ Implementare pertanto una funzione di bubble-sort

Notare come si è passati, dal problema reale, alla sua applicazione virtuale in modo quasi naturale e impercettibile. Per nostra fortuna, alcuni dati reali sono molto facili da trasformare in virtuale.

Questa funzione di bubble-sort ipotizzata, la si potrebbe chiamare "software". Il suo termine tecnico, tuttavia, è "**Configuratore di Prodotto**", perché il suo scopo è soddisfare le necessità di un committente con una soluzione ad hoc per esso

Chiaramente, potremo lasciare a questo configuratore di prodotto un qual certo livello di personalizzazione. Ad esempio, potremmo lasciare il bubble-sort come modello "standard" di ordinamento e, nel contempo, implementare altri algoritmi di ordinamento perché, magari, ora abbiamo più tempo per farlo. Oppure, potremo permettere di eseguire, tramite comoda selezione di opzioni, ordinamenti non necessariamente per ordine alfabetico. L'importante, e che si vuol provare, è che un Configuratore di Prodotto è sempre personalizzabile.

2) MTG – Magic: The Gathering

2.1) Introduzione

Figura 1: Logo di Magic: The Gathering

Magic: The Gathering [1] (italianizzato in "*Magic: L'adunanza*" fino al 2003, e formalmente conosciuto con le abbreviazioni "MtG" o "Magic") è un gioco di carte collezionabili, creato da Richard Garfield e poi pubblicato dalla Wizard of the Coast, nel 1993. È anche il primo gioco di carte collezionabili del mondo, e qualsiasi altro cosiddetto TCG, come Pokemon, Yu-Gi-Oh!, Harry Potter e simili, necessariamente derivano da lui

Durante una partita di Magic, i giocatori incarnano dei maghi, o meglio, dei Planeswalker ("viandanti dimensionali") che si sfidano all'ultimo sangue congiurando creature e lanciando incantesimi e contro-incantesimi, tutti contenuti nel "grimoiro", ossia un mazzo di carte di Magic.

Caratteristica poi rimarcata sempre di più nei vari altri giochi di carte, la composizione di un grimoiro è completamente libera – è necessario solo sottostare ad alcune regole sulla quantità di determinate carte al suo interno - permettendo di fatto la totale personalizzazione e, spesso, caratterizzazione del mazzo. Anche le condizioni di vittoria sono più di una: sebbene la più usata è quella di ridurre a 0 i Punti Vita avversari (i giocatori partono con 20 a testa) tramite incantesimi e attacchi di creature, si può anche vincere per esaurimento grimoiro avversario, oppure con condizioni

[1] http://logonoid.com/magic-the-gathering-logo/

più esotiche come l'uso di Segnalini Veleno, o condizioni di potenti magie soddisfatte.

Non esiste mago senza magia (sebbene esistono, paradossalmente, carte che possono essere giocate senza "potere magico"), e un Planeswalker può ottenere il suo potere magico, altresì detto "Mana", principalmente da carte apposite come le Terre, anche esse facenti parte del suo Grimoiro.

Figura 2: Loghi delle prime espansioni di Magic: The Gathering

Se già tutto questo può indurre i più profani in confusione [2], avverto che non abbiamo che varcato la soglia di Magic – e di qualsiasi altro gioco di carte! Queste meccaniche basilari impallidiscono di fronte alla mole di dettagli specifici delle carte quali categorie, tipi ed effetti – e neanche si accennerà alle regole ed istruzioni tecniche che tengono in equilibrio questi giochi: manuali così vasti e in continua espansione da richiedere la figura di giudici di diverso rango nei tornei più importanti. Fortunatamente, gli aspetti base del gioco diventano abbastanza facili da apprendere, comprendere e memorizzare dopo alcune partite: la figura del giudice scongiura i giocatori dall'obbligo di dover tenere a mente le varie connessioni tra questa e quest'altra abilità, in determinate situazioni di gioco.

[2] http://mtg-realm.blogspot.it/2015/02/tumblr-tuesday.html

2.1.1) Focus: Uno Stile, Un Colore

La magia ha tanti colori diversi – pertanto in Magic assume i colori indicati dall'alchimia: bianco, rosso, blu, verde e nero (sebbene, recentemente, è stato introdotto il "colore Incolore"). Ogni colore rappresenta un certo aspetto con forze e debolezze particolari:

Figura 3: Il simbolo di Mana Blu

Blu : "*Il mondo è un'opportunità, che solo il saggio può trasformare in realtà*"... il Blu [3] è il colore di logica e tecnologia, che ricerca la perfezione attraverso la conoscenza. Crede che l'esistenza non abbia un senso di sua natura – sta al Planeswalker Blu imporre la sua volontà in essa, modellando la realtà come fosse argilla, dichiarando la propria volontà come nuova ragione di esistenza. Uno sforzo, chiaramente, raggiungibile solo tramite l'erudizione e la ragione, e non tramite l'introspezione o le emozioni

L'amore per l'erudizione, pertanto, permette a un Blu di poter pescare molte carte (la ricerca di una risposta nel proprio grimoiro), mentre la ragione permette di poter contrastare gli incantesimi altrui (conoscenza dell'essenza magica, desiderio di cambiare il mondo). Un colore molto mentale, che gli permette, seppure con minime capacità, di distruggere il mazzo avversario, o indebolirne le creature (assalti mentali).

[3] http://mtgsalvation.gamepedia.com/Blue

Le sue magie e creature non sono potenti ma sono difficili da catturare o hanno la capacità di volare (ricerca di efficienza, e suggerimento del tipo "non immischiarti nelle faccende dei maghi"), le poche eccezioni a questa regola hanno fatto la storia, e si sono rivelate le più letali carte del gioco.
La debolezza fisica delle creature, combinata alla generale incapacità di infliggere danni diretti, rende un Planeswalker blu relativamente innocuo rispetto ai suoi simili

Figura 4: Il simbolo di Mana Verde

Verde : *"Il mondo è della natura, dove il rispettoso prospera e l'arcigno trova sciagura"*... il Verde ([4])è il colore dell'istinto e dell'indipendenza, che ricerca l'accettazione attraverso la crescita
Vive in armonia con la natura, e crede che tutto può essere risolto al meglio lasciando fare tutto a lei, nel tempo. Non imporrà mai la propria volontà sugli altri, eccezion fatta contro la tecnologia, ma se attaccata si difenderà scatenando creature selvagge e ferali che contano solo sulla forza bruta e istinto di sopravvivenza. E se il Planeswalker verde cade in battaglia, ha comunque fatto la sua parte nel ciclo naturale, prendendo la parte del debole che soccombe davanti al forte.
Il Verde, pertanto può contare su un buon potere di pesca di carte (istinto, proliferazione), ma anche su un buon potere curativo (rigenerazione naturale) ed ha anche molti modi di generare mana

[4] http://mtgsalvation.gamepedia.com/Green

in fretta (abbondanza della natura). Può addirittura contare su magie che non possono essere contrastate (dominio della natura)
Un Verde può anche congiurare un gran numero di creature (simbiosi naturale, potere nel branco), alcune grandi, che Travolgono quel che trovano con la loro mole imponente, alcune piccole che si rifanno con abilità naturali come la cattura di creature volanti oppure la distruzione degli artefatti (odio per l'artificiale), oppure ancora l'Attrazione e il Veleno per sconfiggere i più temibili
Purtroppo, le sue creature sono o troppo grandi e costose da evocare, oppure con abilità troppo insipide o esotiche

Figura 5: Il simbolo di Mana Bianco

Bianco : "*Il mondo è una comunità, conformati e vivi nella pace e nella bontà!*" …il Bianco [5] è il colore della morale e dell'ordine, che ricerca la pace attraverso la struttura
Tipicamente burocrate e bigotto, si ritiene portatore di equilibrio, e al contempo superiore a chiunque non condivide le sue idee o chi discute i suoi dogmi, incluso sé stesso: la fede di un Planeswalker Bianco è incrollabile e inflessibile, e portata a un singolo scopo: pace attraverso la difesa… eventualmente trasformata in attacchi preventivi, aiutati dall'alto.
L'amore per la difensiva permette a un Bianco di Proteggere sé stesso e le sue creature dagli altri (divina provvidenza), e di

[5] http://mtgsalvation.gamepedia.com/White

guadagnare abilità mentre il nemico attacca (giustizia), ed eccelle nel curarsi (risanamento divino), o distruggere ed Esiliare (ira divina)

Le sue creature trovano potenza nel gruppo, e i singoli possono trovare ulteriore potenza in equipaggiamenti e incanti (Cavaliere divino in armatura splendente).

Tuttavia, le sue creature sono deboli se prese singolarmente, e se non vengono protette adeguatamente.

Figura 6: Il simbolo di Mana Rosso

Rosso : "*Il mondo è un'esperienza, di cui le emozioni sono l'essenza*"

…il Rosso [6] è il colore dell'impulso e del caos, che cerca la libertà attraverso l'azione

Crede nel provare costantemente novità casuali, che causano eccitazioni dei sensi e nuove emozioni – l'essenza stessa della vita - e che questa sua continua ricerca non deve essere ostacolata da niente e nessuno. Un Planeswalker Rosso pertanto odia le regole, e chiunque cerchi di ostacolare la libera espressione dovrebbe essere zittito con veemenza. Una mentalità, questa, che porta parecchi problemi a questi Planeswalker… ma di cui in genere non si preoccupano troppo: è esperienza anche quella, d'altronde!

L'amore per l'impulso fa spesso pescare per poi scartare subito altre carte (potenziale su presente) e produce mana da usare subito (attimo di eccitazione), e un Rosso conta sul più potente arsenale di

[6] http://mtgsalvation.gamepedia.com/Red

magie che esiste per danneggiare direttamente l'avversario (Emozioni brucianti) ma anche se stesso (caos).
Le creature di un Rosso sono spesso veloci e con attacchi avventati (furia), e capaci di potenziare il proprio attacco (emozione pura)
Tutto questo viene spesso ripagato con un basso istinto di autoconservazione: sovente un Planeswalker Rosso infligge danni anche a sé stesso, oppure le sue creature hanno capacità autodistruttive o non provvedono affatto a difendere il loro maestro

Figura 7: Il simbolo di Mana Nero

Nero : "*Il mondo è potere, che solo io devo possedere*" …il Nero [7] è il colore del parassitismo e dell'amoralità, che cerca potere attraverso opportunismo e spietatezza.
Onnipotenza ed egocentrismo alimentano un Planeswalker Nero, e vuole ottenere potere ad ogni costo, anche andando contro le morali e il buon senso, senza contare sull'aiuto di altri. Abomini, omicidi, furti e patti con forze oscure: ogni mezzo è valido per questi egotisti superbi, che si credono superiori a tutto e tutti. E se qualcosa non può essere ottenuto, allora verrà annullato affinché non lo possano avere nemmeno gli altri.
Un Nero seguo uno dei principi alchemici più famosi, dello Scambio Equivalente: può liquefare le proprie risorse per trasformarle in altro (dare qualcosa per riceverne di egual valore). Alternativamente, è tipico distruggere la mano dell'avversario o

[7] http://mtgsalvation.gamepedia.com/Black

derubarlo della sua stessa forza vitale (parassitismo). Inoltre, è solito fiaccare o uccidere le creature avversarie (omicidio)
Le creature di un Nero sono non-morte, iconicamente, a rappresentare il potere sopra la morte (negromanzia), e sono spesso dotate di agilità e Primi Attacchi (tecniche sporche)
Un Nero ha solo una singola, pesante debolezza: l'incapacità di controllare ciò che non è reale. Pertanto, spesso non sa come difendersi dagli incantesimi avversari

Figura 8: Il nuovo simbolo di Mana Incolore

Incolore : *"Non sono di questo mondo"* …L'Incolore [8], in principio, non era un colore vero e proprio, ma recentemente sono nate carte il cui costo può essere pagato solo da questo tipo di mana. Il "colore Incolore" viene utilizzato dagli Eldrazi, esseri amorfi e ancestrali senza forma fisica o allineamento, che vivono al solo scopo di cibarsi di un Piano dimensionale, drenandolo di tutto il mana fino alla distruzione totale se necessario. Essendo pertanto nemici ad ogni piano dimensionale, l'uso di un colore che non appartiene ad essi risulta quantomeno appropriato.

2.1.2) Focus: I formati

Uno dei punti di forza di Magic consiste nell'uso di Formati che costringono i giocatori a poter usare solo determinate carte, e

[8] http://magic.wizards.com/en/articles/archive/feature/oath-gatewatch-mechanics-2015-12-28

quindi eventualmente ad aggiornarsi periodicamente. Questi sono alcuni dei vari Formati finora creati:

- **Vintage, o Tipo 1**: valgono tutte le espansioni (eccetto Unhinged e Unglued), presenta una F&L (Forbidden & Limited), ovvero una lista di carte che non possono essere usate, o possono essere incluse al più in una copia;
- **Legacy, o tipo 1,5**: valgono tutte le espansioni. Non esistono carte limitate, ma un gran numero di esse sono proibite;
- **Standard, o Tipo 2**: quello più giocato, vale l'ultimo Core Set, più le ultime due espansioni;
- **Modern**: è il più recente, e accetta solo le carte stampate dopo il 2003. Sostituisce l'Extended;
- **Block**: sono valide solo le carte di un certo set o blocco, generalmente l'ultimo;
- **Commander, o Elder Dragon Highlander**: un formato in cui si seguono regole notevolmente diverse: Il mazzo deve essere composto da esattamente 100 carte, di cui una è il Generale, e non può avere un Sideboard. In aggiunta, è possibile includere una singola copia di ogni carta (eccezion fatta per le terre base), e si possono usare solo carte che condividono almeno un colore con quelli del Generale;
- **Pauper**: Valgono solo le carte comuni uscite da Mirage in poi. Questo formato nasce online ed è stato concepito come un'alternativa economica al gioco normale, e da lì ha preso anche piede nel cartaceo. La Wizard of the Coast non ha ancora riconosciuto ufficialmente questo formato;
- **Limited**: includono i formati Sealed Draft, Booster Draft e Rochester Draft (non più ufficiale): il giocatore è costretto a creare un mazzo direttamente sul posto, usando solo carte trovate in bustine in genere dell'ultima espansione uscita, e poi giocare con quello.

2.2) Breve Storia di Magic: The Gathering

2.2.1) I primi passi – Resoconto di Richard Garfield, 1993

Figura 9: Immagine della scatola del gioco da tavolo Cosmic Encounter

Nell'anno 1982, Richard creò un gioco di carte chiamato Five Magics. L'ispirazione di tale gioco veniva da un altro gioco da tavolo, Cosmic Encounter [9], dove circa 50 razze aliene, ognuna con i propri vantaggi e peculiarità, si sfidavano a vicenda per conquistare un pezzo di universo. Le combinazioni che il gioco permetteva erano "*non completamente prevedibili, ma nemmeno completamente sconosciute, come certe forze che capisci e non capisci allo stesso tempo*".

Nel 1992, Richard ebbe il suo primo contatto con la Wizard of the Coast. In un primo momento, voleva proporre un certo gioco da tavolo, tale RoboRally, ma la Wizard, rappresentata al tempo da Peter Adkison e James Hays, rifiutò tale proposta, indicando di non

[9] https://boardgamegeek.com/boardgame/39463/cosmic-encounter

essere pronti ad un gioco da tavolo. Piuttosto, espressero il desiderio di avere "*un gioco che poteva essere giocato rapidamente e con minimo equipaggiamento, che sarebbe andato bene ad ogni convention*". Richard pensò in un primo momento a Safecraker, un altro gioco di carte di sua invenzione del 1985. La scelta poi ricadde sul concetto di Five Magics.

Figura 10: Allegoria di Richard Grafield usata nella carta "Richard Garfield, Ph.D"

Durante la progettazione, i dubbi di Richard [10] spaziarono sia dentro che fuori il gioco. Uno dei primi problemi, uno standard, fu quello di evitare la creazione di carte inutili: "*perché creare carte con cui non giocheranno mai?*", pensava. Un altro grosso problema fu combattere la "sindrome del bambino ricco": "*cosa può trattenere qualcuno dall'uscire, comprarsi dieci mazzi potenti ed essere imbattibile?*". Fu allora che costrinse chi giocava all'ormai ufficializzato Magic: The Gathering a dover scommettere una carta. "*Se il gioco richiedeva parecchia abilità, allora il giocatore che comprava potere sarebbe stato una preda facile per i giocatori […] E naturalmente ci sarebbe stata la sensazione che comprare tante chip*

[10] https://alchetron.com/Richard-Garfield-559999-W

da poker non ti rende un vincitore". Per sua fortuna, Magic si rivelò un gioco divertente, pertanto questo problema si avvertì ben poco.

Figura 11: Logo delle espansioni Alpha (A) e Beta (B)

Nel 1993 pertanto nacque la prima pubblicazione, Alpha [11], composta da 120 carte. L'idea era comprare uno di questi set, dividerli in due mazzi da 60 (la dimensione minima, tuttora, di un mazzo di Magic), e poi giocare con essi, scommettendo una carta ogni volta. Fu allora che Richard capì che poteva esserci spazio anche per altri tipi di espansioni: avrebbero creato combinazioni divertenti e sorprendenti… forse anche troppo: era necessario infatti, in tal caso, tenere a bada l'equilibrio del gioco, onde evitare che una combinazione di carte troppo potenti potesse monopolizzare le partite. Le due espansioni successive, Beta [12] e Gamma, assestarono questo equilibrio e permisero la nascita di reputazioni tra i giocatori. Si badò anche a non rivelare il contenuto del set, affinché i giocatori rimanessero in allerta durante tutta la partita per evitare carte sconosciute e a cui non avevano una pronta risposta. Già da allora iniziarono a scomparire le carte che prendevano il controllo delle creature dell'avversario – forzavano infatti i giocatori a suicidare i propri servitori onde evitare di vederseli rivoltati contro al turno avversario. Richard capì che "*nessuna carta dovrebbe veramente essere a rischio a meno che uno non decida di scommetterla con l'avversario*".

[11] http://www.mtgsalvation.com/forums/creativity/artwork/340334-custom-set-symbols
[12] http://www.mtgsalvation.com/forums/creativity/artwork/340334-custom-set-symbols

Figura 12: La carta "Counterspell" edizione Alpha

I tentativi di evitare le combinazioni "troppo sorprendenti" alla fine fallirono [13], com'era naturale che sia: il maggior numero di carte permetteva combinazioni sinceramente impreviste anche dallo stesso Richard, e dai risultati semplicemente degenerati e frustranti. Richard non corse sempre ai ripari in questo caso, poiché notò che tali mazzi si rivoltavano contro i loro stessi creatori: o li privavano della competizione, dato il loro strapotere, o li privavano di sfidanti, stufi di andare sempre contro a sconfitta certa. Fu da allora che nacquero i mazzi "a tema strano", che per lo più puntavano a divertimento o stile. Ci sono stati, chiaramente, mazzi a tema strano degenerati, come il Mazzo della Ricorsione Infinita, ma di questo lo stesso Richard parla poco volentieri. Era però chiaro che "*ogni mazzo di ogni giocatore è come un personaggio: ha la sua personalità e segreti, spesso hanno dei nomi…*"

Dopo due anni, Magic iniziò anche ad avere anche un'economia di mercato aperta, dovuta alle cattive capacità di prezzamento, per lo più soggettive, dei giocatori. Non era raro incontrare giocatori che

[13] http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=102

compravano carte da tutti bistrattate, generare con esse una combinazione veramente potente e poi fare grossi guadagni rivendendo la "ricetta" del suo mazzo low-budget.

Con il numero di carte in aumento, la nascita di mazzi personalizzati e/o basati su combinazioni fece capire a Richard che poteva esserci un filo che collegava tutti i mazzi che ogni giocatore creava. Fu allora che concepì Magic come un "multiverso", e gli allora maghi come "planeswalker". In un multiverso ci sono molte dimensioni e universi, ognuno con le proprie regole, pertanto se le interazioni tra di essi erano "strane", non lo sarebbero state in un simile ambiente aleatorio. Così facendo, inoltre, i nuovi set di carte si sarebbero tranquillamente accomodati nel multiverso – sarebbe stato come aggiungere un'isola in un oceano sconfinato.

Redatto nel 1994, fu infine nel 2 settembre del 1997 che venne approvato il brevetto con seriale 5.662.332, di Magic: The Gathering, che vide come creatore Richard Garfield e come assegnista la Wizard of the Coast, e al cui interno venivano siglate le regole essenziali e non del gioco. Quella che segue è l'Estratto di tale brevetto ([14]):

[14] http://www.google.com/patents/US5662332

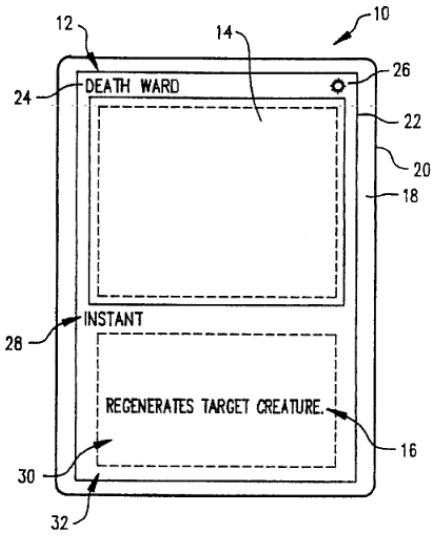


Figura 13: Template di una carta di Magic, come mostrato nel Brevetto

"Il qui presente è un nuovo metodo di gioco e componenti di gioco nella forma di un gioco di carte collezionabili […] Tuttavia, i componenti del gioco possono assumere altre forme, come giochi da tavolo, o giochi giocabili in altri tipi di supporto, come giochi elettronici, videogiochi, giochi per computer e network interattivi.

In questa versione, i componenti del gioco comprendono carte di energia o mana e carte di incantesimi o comandi, aventi comandi e incantesimi associati al loro interno che usano l'energia per permettere a un giocatore di attaccare, difendere e modificare gli effetti di altre carte mana, carte magia e le fondamentali regole del gioco.

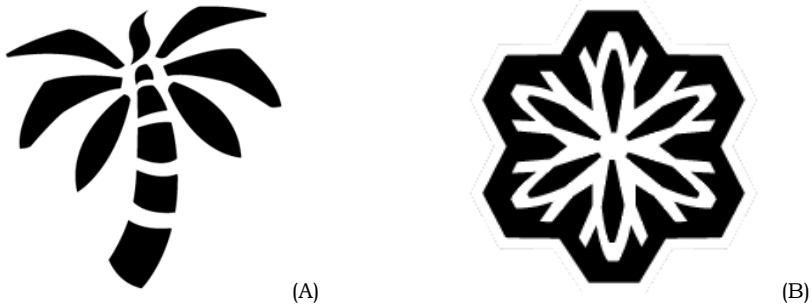
Obiettivo del gioco è ridurre i punti vita degli altri giocatori a un valore inferiore a 1. In questo gioco di strategia e fortuna, i giocatori creano la loro libreria di carte, preferibilmente di carte collezionabili, e giocano la loro libreria o mazzo di carte contro altri mazzi di carte di giocatori avversari. Le carte sono ottenibili da rivenditori autorizzati, scambiate con altri giocatori o collezionisti, e carte vinte a giochi e tornei".

2.2.2) L'Evoluzione – Resoconto di Richard Garfield, 2003

Figura 14: Logo dell'espansione *Fallen Empires*

Magic era nato, e doveva rimanere, un gioco di carte collezionabili – o "scambiabili", come preferisce dire Richard. "*i buoni giochi rimangono per sempre, le collezioni vanno e vengono.*" Purtroppo per Magic, ci fu un periodo di grande speculazione economica, che minacciava seriamente la percezione dello stesso come gioco piuttosto che come collezione. Prezzi altissimi di set che dovevano essere venduti a quattro soldi potevano fare seriamente male al numero di nuovi giocatori. Questo periodo durò fino all'uscita di Imperi Caduti [(15)], la quinta espansione, che fece collassare tale mercato speculativo: le carte erano così tante che per la base di giocatori era impossibile decretare da subito quale carta stava bene e in quale combinazione. Pertanto, il tempo "perso" permetteva ai giocatori di mettere le mani, con un po' di fortuna, sulle carte giuste al momento giusto.

(15) http://www.powernine.com/sets/105.html



(A) (B)

Figura 15: Loghi delle espansioni Mirage (A) e Ice Age (B)

La grande mole di carte però portò un nuovo problema: il cosiddetto "power creep", ossia una richiesta di carte sempre più potenti e nuove che potessero competere con le precedenti, e spodestarle dalla loro posizione dominatrice. Fu con l'introduzione di Ice Age [16] e Mirage [17] che Richard iniziò a concepire il concetto dei formati di gioco. Nei piani originari, Ice Age doveva terminare "Magic: The Gathering" per poter far iniziare "Magic: Ice Age" (anche perché Richard rabbrividì nel sapere che, di tutto Ice Age, solo 2 carte erano effettivamente considerate utili dai giocatori). L'idea era generare un nuovo card pool che ri-determinasse la posizione di potenza di certe carte, nonché permettesse ai giocatori più antiquati di poter comunque giocare al "solito gioco" senza sentirsi costretti a comprare altro. Ma ci furono vibranti proteste alla divisione di Magic in due "giochi", pertanto Richard ebbe l'innovazione di creare i cosiddetti "formati". I Formati permettevano di giocare a Magic, invero, ma solo utilizzando determinati pool di carte. Il più famoso allo stato attuale è il cosiddetto "Standard", dove sono considerate legali solo le carte facenti parte degli ultimi 2 set e l'ultimo set Core. Questo costringe un giocatore a dover aggiornare periodicamente il proprio mazzo – nonché ai nuovi giocatori di poter entrare nel gioco più o meno con gli stessi strumenti dei veterani- ma al contempo si assicura che una carta possa mantenere la propria posizione di potenza solo per

[16] http://www.powernine.com/sets/106.html
[17] http://www.powernine.com/sets/109.html

circa 6 mesi (le espansioni sono a cadenza trimestrale), per poi non essere più giocata – almeno a Standard.

Figura 16: La carta "Counterspell", edizione "Jace vs Chandra" (2008)

Un'altra battaglia con cui Magic si scontra al giorno d'oggi è l'essere veloce. O, quantomeno, avanzare dai primi giorni in cui una partita poteva durare anche ore – è chiaro che invece una partita di carte non dovrebbe durare più di mezz'ora. Ciò non significa necessariamente che i giocatori devono ricordarsi a menadito ogni singola combinazione di carte, ma quantomeno si doveva evitare, in principio, di generare situazioni di gioco che avrebbero bloccato una partita inutilmente. (18)

(18) http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=185820

Figura 17: Logo del Magic: The Gathering Pro Tour

E, infine, una volta che Magic iniziò a diventare famoso e con basi ben solide, iniziarono i Pro Tours [19], ovvero delle competizioni con "*grossi premi in denaro, tali da permettere a qualcuno, se abbastanza bravo, di viverci, di Magic*". Le ombre di dubbio sollevate dalla Wizard sulla proposta si dissiparono in fretta, come videro che molti giocatori iniziavano ad affinare le loro abilità al fine di poter competere meglio.

L'ultima frontiera è rappresentata dal porting online di Magic, nel 2003. In un primo momento, Richard temeva che il gioco online potesse rimpiazzare quello cartaceo, pertanto si utilizzò il modello di vendita di carte virtuali: un giocatore si registra e poi compra, in moneta sonante, la possibilità di inserire nel proprio mazzo virtuale certe carte (Chiaramente, in un secondo momento poteva "redimere" tali carte ottenendone una copia vera e cartacea).

[19] http://www.italianmagicjudges.net/index.PHP?p=articoli&id=1952

2.3) Lo Sviluppo Online

2.3.1) MTGO – Il Futuro di Magic?

Figura 18: Logo di Magic: The Ghatering Online

MTGO [20] ha raccolto, nel tempo, molti giocatori, per lo più formati da una base di veterani. La ragione, è facile da vedere:

- La non-fisicità delle carte permette di risparmiare qualcosa in termini di bustine protettive e porta-mazzi
- Si può giocare con chiunque, anche di altri paesi, restando seduti comodamente a casa propria. Bastava darsi un'ora per l'appuntamento, e una connessione internet
- Alcune funzioni di gioco automatizzate permettono di risolvere le controversie più semplici sulle regole delle carte senza infastidire giudici
- Comunque sia, permetteva di poter comprare carte a un prezzo ragionevolmente fisso, e da un rivenditore di fiducia
- La possibilità di comprare "per carta" e non "per pacchetto" permetteva di poter avere accesso anche a carte appartenenti a espansioni o cores di difficile reperibilità, permettendo così l'accesso a formati Vintage e Legacy più agevolati
- Si potevano scalare le classifiche, anche ufficiali, e farsi una carriera senza necessariamente andare in locali ed eventi reali

[20] http://magic.wizards.com/en/articles/archive/building-budget/magic-online-organized-play-update-2008-09-18

La presenza di decklist online rende anche la schedatura che in genere si fa ai tornei una questione triviale, e accedere al contenuto di un mazzo diventa altrettanto facile. Non c'è da stupirsi pertanto se sono nati dei servizi online che mantengono un database di queste "ricette"…

2.3.2) Il Caso: MTGTop8.com

Figura 19: Logo di MTGTop8.com

MTGTOP8.com [21] è uno di questi siti. Il suo scopo è semplice: raccogliere informazioni su eventi (il più vecchio del 2007) e registrare le ricette, o decklists, dei primi 8 mazzi vincitori (da qui il "top8"). A volte, se gli è permesso, il sito cerca di dare informazioni anche di più mazzi dello stesso evento, non necessariamente in top8, altre volte le informazioni sono così scarse che a malapena si sa chi è il vincitore di un evento.

[21] http://mtgtop8.com/

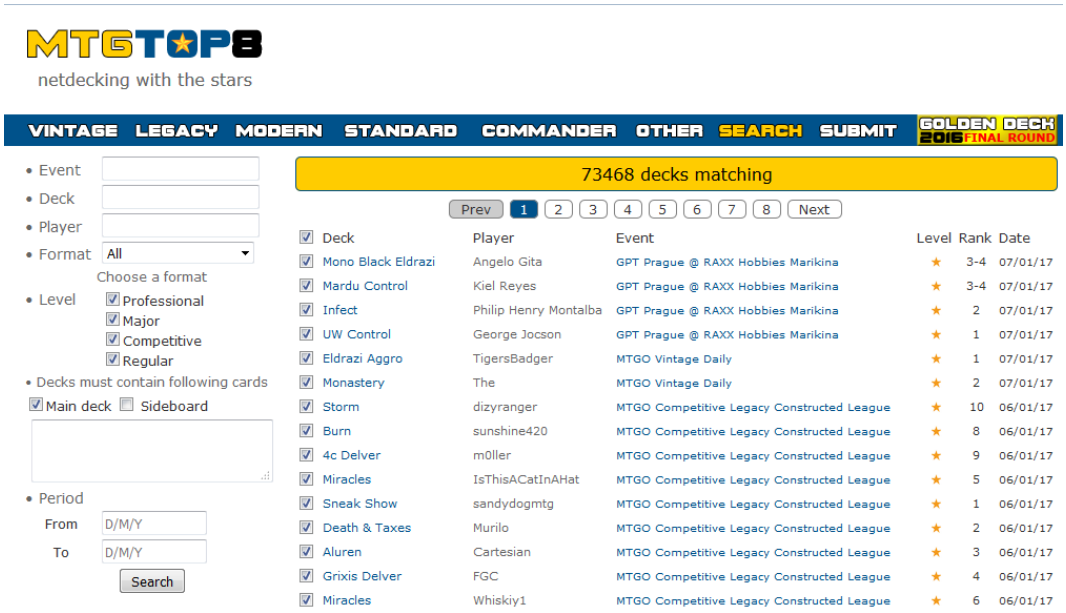


Figura 20: Schermata di ricerca del sito MTGTop8.com . Le ricerche sono filtrabili per eventi, mazzi e giocatori, o si può addirittura ricercare mazzi con specifiche carte o appartenenti a un certo periodo di tempo

Il sito offre anche altre funzioni interessanti: consente di ricercare [22] i giocatori che hanno preso parte a determinati eventi, o i mazzi giocati da un singolo giocatore. Permette anche di poter vedere in quali mazzi una certa carta è stata giocata. Poiché divide gli eventi in formati, è anche in grado di fornire statistiche su quale tipo di mazzo è più usato, e quale è la top10 delle carte più usate in ogni formato diverso.

Statistiche, statistiche, statistiche, ma cosa si può fare con tutto questo? Indubbiamente, può aiutare un giocatore veterano a vedere quali sono le carte che vanno più in voga (concetto noto come “meta”) e trovare come rispondere ad esse; può aiutare gli storici a vedere lo sviluppo di certe carte, ricostruirne la cronistoria e vederne periodo di ascesa e declino. Ma permette anche un’altra interessante funzione…

[22] http://mtgtop8.com/search

2.3.3) Il concetto di Netdecking

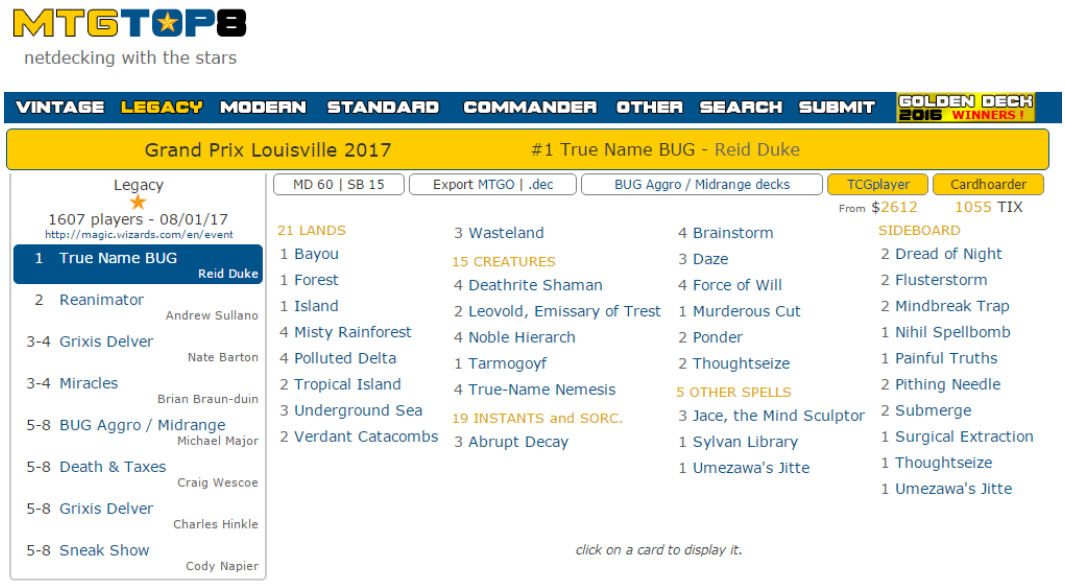


Figura 21: Schermata di MTGTop8.com che mostra una decklist registratavi

Il motto di MTGTOP8 è: "Netdecking with the Stars" [23]. L'operazione di Netdecking è tipica di qualsiasi gioco di carte, e consiste nel prendere ispirazione da alcune decklist trovate online ("*net*") per poter generare un nuovo mazzo ("*decking*").

Nei casi più candidi, questo viene fatto per istruirsi su come certe carte possano stare bene in combinazione con altre. In casi un po' più "biechi", questo permette un'opera di "cookie cutting", ovvero la creazione di un mazzo esattamente identico a quello visto (fatto, per l'appunto, "a stampino"), azione eseguita seguendo l'adagio "*se lui ha vinto con quel mazzo, posso farlo anche io!*".

In genere, chi segue questa filosofia scopre ben presto che è inutile avere l'arma più potente del mondo in mano se non la si sa usare – in altri termini, non ha perso nemmeno un secondo a studiare le macro-combinazioni di carte che hanno reso quel mazzo vincente, nell'illusione che questi disponesse di una "modalità auto-pilota"

[23] http://mtgtop8.com/event?e=14415&d=286245&f=LE

che facesse tutto da sé. I cookiecutter più seri in genere hanno la decenza di sapere con cosa giocano, e a volte riescono a raggiungere gli obiettivi prefissati fondendo comprensione del mazzo ad effettiva potenza delle sue combinazioni.

3) Il Configuratore – Magic Academia

Indipendentemente dalla buona fede del suo utilizzatore, si desidera comunque creare un ambiente in cui fare un po' di netdecking… con un pizzico di statistica che aiuta.

Il configuratore pertanto si dirama in questi componenti base:

- Uno strumento che permetta di scaricare mazzi da MTGTOP8.com
- Uno strumento che permetta di creare mazzi, basandosi sui mazzi che sono stati appena scaricati

3.1) Introduzione – Index.PHP

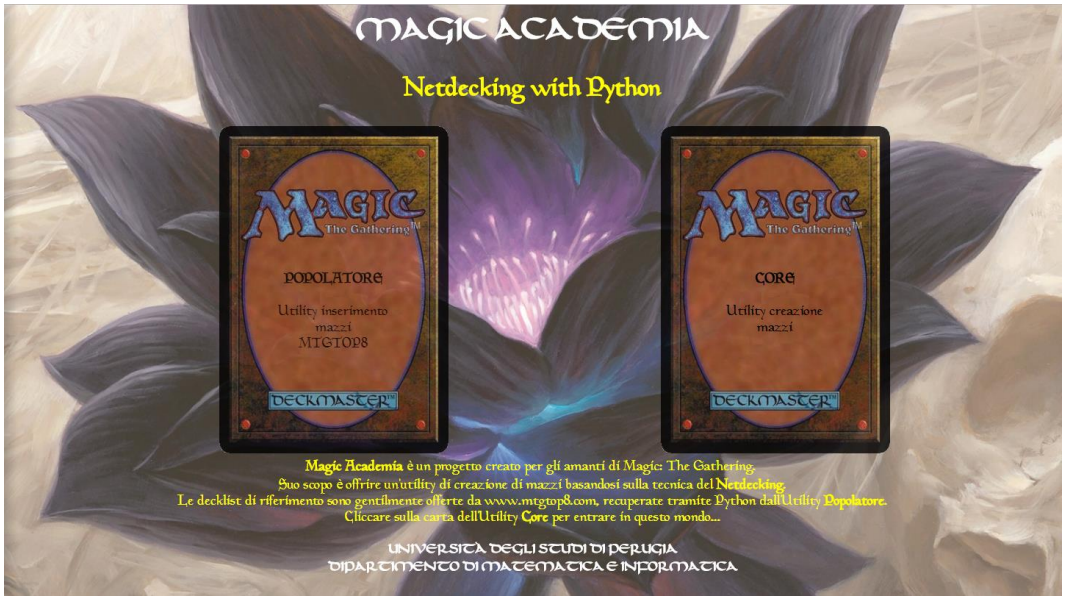


Figura 22: Schermata di Index.php

La pagina principale di questo configuratore ([24]) è abbastanza semplice nella sua natura. Non ha altro che un titolo, una presentazione sommaria e due link sotto forma di carte.

[24] http://mtgweb.dmi.unipg.it/mazziapp/magicacademia/index.php

Contrariamente a quanto uno potrebbe aspettarsi, index.PHP, la pagina principale, non è stata la prima delle pagine create, bensì l'esatto contrario. È stato un po' come partire da una soluzione e dedurne poi l'algoritmo che l'ha generata via ingegneria inversa: prima si sono create le funzioni, poi una pagina che le raccogliesse tutte.

Essendo pertanto l'ultima, è anche quella con gli effetti grafici più carini, vista la notevole compattezza del codice sorgente. Questa è una pagina più HTML che PHP, anzi, ad essere precisi è solo una combinazione di tag HTML e un singolo script che, volendo, poteva essere ricreato nello stesso css

3.1.1) Uno sfondo semitrasparente

Una cosa che potrebbe saltare all'occhio è stato l'uso di uno sfondo che si ripete, e che è anche semitrasparente. Purtroppo, sebbene i css permettano di poter inserire un'immagine in un div e poi da lì modificarne il valore opacity, questa feature non è permessa con le immagini di sfondo. Pertanto, l'immagine semitrasparente è stata ritoccata per avere tale effetto – un lavoro che fortunatamente software freeware quali paint.NET e Gimp riescono a svolgere in relativa facilità.

3.1.2) Da una carta che si colora al reverse-opacity

I link ai due strumenti sono generati sotto forma di dorso di carta – anch'esso, per la cronaca, fotoritoccato per togliere le 5 sfere di mana tipiche. Ora, le carte seguono un effetto "reverse-opacity", ovvero sono trasparenti di natura e, al passaggio del cursore, vengono rese opache come al naturale. In principio, tuttavia, si

voleva generare un effetto highlight al passaggio del mouse, che avrebbe dato una sorta di "aura" colorata alla carta.

In un primo momento, si è tentato di usare la funzione box-shadow, il cui funzionamento, fu osservato per la prima volta su alcuni paragrafi che venivano, effettivamente, illuminati nella maniera desiderata al passaggio del mouse. Dopo alcuni test, si è compreso che per "ombra" si intendeva letteralmente un'ombra (piuttosto che un "adombramento") proiettata dal div, o comunque dall'oggetto, e che pertanto non ricopriva anche l'oggetto in sé.
Si era ancora nella fase "colora-la-carta", pertanto c'erano due opzioni:

- Lasciare la carta "com'è" e, passando sopra il cursore, generare un'alone-ombra colorato
- Colorare la carta quando ci si passa sopra con il cursore, senza aloni

Optai per la seconda scelta, che si rivelò più vicina a quello che volevo fare. L'effetto è facilmente realizzabile: si colora il background del div in cui la carta è contenuta di un certo colore e poi, passando sopra la carta, questa diventa più trasparente, ottenendo pertanto una sfumatura di colore desiderato.

Purtroppo, incappai in un nuovo problema: il div che conteneva il nome del link e la carta stessa, di fatto, contavano come due diversi oggetti su cui si poteva applicare onhover (il fatto che il div del testo era contenuto dentro il div dell'immagine, apparentemente, era insignificante). Il risultato, pertanto, era l'immagine della carta che veniva resa più trasparente, ma il nome del link rimaneva di egual opacità – peggio ancora, passare sopra il nome del link rendeva di nuovo opaca la carta.

Si prospettarono alcune scelte:

- Creare un div (elemento generico HTML) “fantasma” che ricopriva la carta, colorato in modo trasparente. Passando sopra la carta, in realtà si passava sopra questo div, che si rivelava e “velava” la carta con il suo colore, nella sua interezza. Cliccandovi sopra, poi, si sarebbe attivato il link al tool desiderato
- Fotoritoccare la carta aggiungendo il testo all’immagine, generando così una nuova immagine immodificabile
- Creare uno script JavaScript che avrebbe reso trasparente testo e immagine qualora l’onhover si sarebbe compiuto sopra uno di questi due oggetti del DOM

Scartata la prima per bruttezza, e la seconda per pigrizia e amor della portabilità, andai per l’ultima. Una sapiente smussamento degli angoli del div dava l'illusione che era la carta a colorarsi, e non il suo sfondo.

Tuttavia, rimaneva il fatto che lo sfondo aveva un colore, e al caricamento della pagina era possibile vedere, per il tempo necessario a caricare l'immagine del dorso della carta, tale sfondo. Ritenni l'effetto brutto e indesiderato, pertanto optai per il più tranquillo “reverse-opacity”: la carta è trasparente, e al passaggio del mouse essa viene resa di nuovo opaca come di sua natura – il cambiamento a livello di JavaScript fu minimo.

3.1.3) L'uso di una pagina di benvenuto

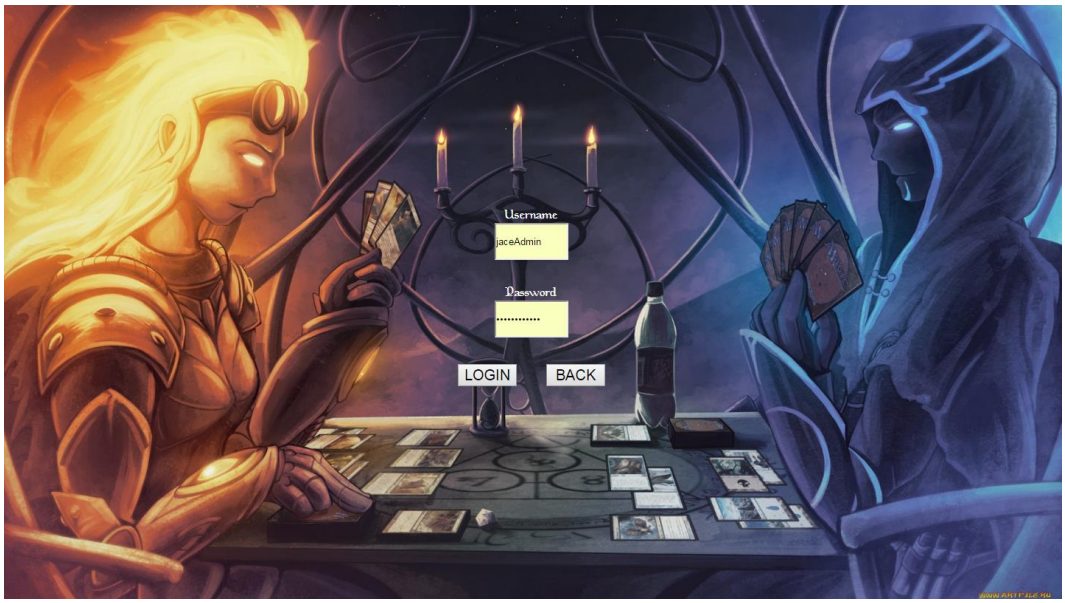


Figura 23: Schermata di login.php

Quando si clicca su uno qualunque dei link, in realtà si viene ridirezionati a una pagina di login ([25]). L'uso di tale pagina si è rivelato necessario per due scopi:

- Per il tool di web scraping: doveva essere permesso l'accesso solo a un admin, vista la presenza, nella pagina, di funzioni in grado di cancellare anche l'intero database con un solo click di mouse
- per il tool di creazione mazzi: quando il mazzo veniva registrato, era necessario sapere il nome dell'utente a cui ricondurre tale mazzo.

La pagina di login è anche essa semplice – nulla più che una form con username e password. Tuttavia, tiene anche a mente, tramite variabile di sessione, in quale pagina l'utente stava andando. Questo serve, nel caso si sta andando verso il tool di web scraping, per negare l'accesso ai non-admin a tale strumento.

([25]) http://mtgweb.dmi.unipg.it/mazziapp/magicacademia/login.php

3.2) Webscraper.php

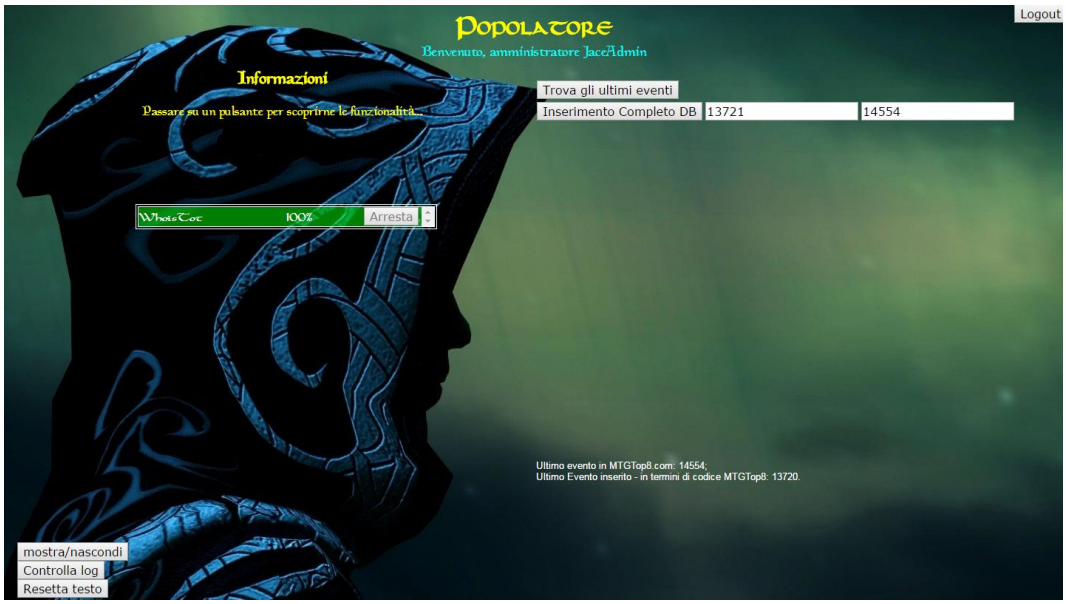


Figura 24: Schermata di webscraper.php

La prima parte, progetto di stage, rispondeva alla richiesta del committente: "*realizzare un tool online che permettesse di scaricare dati da un sito, per poi caricarli in un database. Libertà sui mezzi da usare*". ([26])

([26]) http://mtgweb.dmi.unipg.it/mazziapp/php_test/Popolatore.php

3.2.1) Scelte di Sviluppo

3.2.1.1) L'uso di PHP e Python

Figura 25: Logo di PHP

Il progetto iniziò due anni fa, quando ancora la mia formazione era carente. Era già tanto se sapevo scrivere uno script PHP, e JavaScript e Ajax mi erano completamente alieni. L'unica cosa che avevo appreso con abbastanza facilità era Python, e pertanto con quello andai.

La pagina, pertanto, si basava su PHP [27], che permetteva a un utente di poter eseguire da remoto alcuni script Python, che avrebbero poi eseguito le varie operazioni di web scraping: dalla lettura della pagina fino alla scrittura su database. PHP, di suo, non sfiorava nemmeno il database, fatta eccezione per alcune funzioni ibride come quelle di cancellazione database.

Con il tempo, mi pentii della mia scelta. Quando presentai ai ragazzi dell'Applab questa parte, uno di loro pose in essere la domanda, "ma perché hai usato PHP e Python?". Avevo già previsto questa domanda, e dedicai una diapositiva alla cosa. In essa erano presente alcune parole laconiche quanto oneste: "domanda: perché hai usato PHP e Python? Risposta: perché mi faceva comodo

[27] http://www.virtualars.it/PHP/

allora". con il senno di poi, avrei realizzato una soluzione non-ibrida, full-PHP/JavaScript.

Figura 26: Logo di Python

L'uso di Python [28] mi stava – e sta dando – infatti più problemi che benefici. Uno tra i più importanti: quello di ricorrere ad exec per l'esecuzione remota di script. La decisione di impiegare questa funzione fu accolta con notevoli grattamenti di capo e arricciamenti di naso, vista l'insicurezza che può offrire un comando in grado di eseguire una qualsiasi, deliberata stringa bash. Ma, incapaci di vedere altre soluzioni immediate, optammo per quella. In aggiunta, l'intero progetto si stava svolgendo in quella delicata fase di transizione da Python 2.x a Python 3.x, pertanto l'ambiente Python nativo offerto dal server poteva non essere più adeguato – dovetti aspettare la creazione di una Virtual Machine Python prima di poter tentare qualsiasi operazione da remoto.

Alla fine, il progetto venuto alla luce, e tuttora fruibile, rappresenta una soluzione assolutamente non-ottimale, ma comunque funzionante. Un monito, forse, per i programmatori futuri su come le soluzioni immediatamente disponibili potrebbero non sempre essere le più efficienti.

[28] https://commons.wikimedia.org/wiki/File:Python-logo-notext.svg

3.2.1.1.1) Web Scraping e Python

Come si è detto, bisognava "scaricare i dati da un sito". Questa operazione, in gergo, si chiama WEB SCRAPING.

È un'operazione abbastanza logica e di facile comprensione. Supponiamo di voler ricercare informazioni su un certo argomento. Per esempio, veniamo a sapere che il 95% dei mazzi Vintage gioca un Black Lotus, e vogliamo sapere per quale ragione questa carta è così famosa. Ipotizziamo sia per il fatto che ha un effetto molto potente. Ricercando su internet, ci imbattiamo in un sito di compravendita online, che al suo interno mostra diverse informazioni: immagine della carta, prezzo medio e suo andamento negli ultimi 12 mesi, disponibilità del prodotto, testo, venditori e loro rating di affidabilità, zona di login, barre di ricerca…
Sappiamo che l'effetto compare nel testo di una carta. Pertanto iniziamo a ricercare questo oggetto all'interno della pagina. Alla fine, dell'intera pagina possiamo ignorare quasi tutte quante le varie altre informazioni – indubbiamente utili, ma non per il nostro scopo. Come finalmente leggiamo la sezione del testo, abbiamo trovato la singola e unica informazione che cercavamo.

L'operazione di ricerca e della sezione del testo la svolgiamo noi, umanamente, con occhi che acquisiscono l'informazione e cervello che filtra. Ma come può un computer "leggere" una pagina internet? Come può un computer capire quale informazione è "giusta"? Quest'ultima domanda esula dalle conoscenze richieste per questo scopo – di fatto sfocia troppo sul datamining. La prima domanda, in compenso, può essere risposta con una qual certa facilità:

Quando guardiamo una pagina internet, sappiamo in cuor nostro di star leggendo nulla più che delle linee di codice – anzi, per la precisione ne stiamo leggendo una sola, lunghissima e spezzettata solo per amor di leggibilità. Questa lunga stringa viene poi interpretata dai browser web per formare il cosiddetto albero DOM (*Document Object Model*), dandole una logica simile alla programmazione ad oggetti.
Se pertanto vogliamo cercare un determinato dato all'interno della pagina, esso deve certamente stare all'interno di questo albero DOM. Poiché questo tipo di albero non è ordinato (o quantomeno, non lo è il sotto-albero figlio di Body, dove molto probabilmente sta l'informazione cercata), cercheremo l'informazione ("navigheremo il DOM") "umanamente", identificheremo il nodo che ci interessa e risaliremo verso HTML generando così un cammino.

Il cammino così trovato prende il nome di XPath, ed è in questo modo che un pc esegue web scraping nella sua forma più basilare: la lettura completa di una pagina web.

3.2.1.1.1.1) Lettura del DOM – Da pagina ad albero

Ricapitolando, pertanto, l'operazione di web scraping si esegue in questi passi:

- Leggere umanamente il DOM e ricavare altrettanto umanamente una XPath valida
- Dare alla macchina l'XPath desiderato, e la pagina di provenienza
- La macchina a questo punto "legge la pagina internet"
- E successivamente esegue l'XPath, per poi ritornare il dato desiderato

Il penultimo punto è personalizzabile: sebbene la lettura di un XPath è abbastanza standardizzata, la lettura della pagina internet è più soggettiva. Un po' come quando, di un lavoro, ci interessa il risultato e trascuriamo il metodo con cui si è giunti ad esso

Per Python, che è la parte che fa anche web scraping, si prospettava come prima soluzione una libreria di nome Beautifulsoup. Tale libreria aveva la facoltà di prendere la pagina e restituire il codice HTML, per giunta indentato al fine di renderlo gradevolmente più leggibile. Tuttavia, si è desiderato ricercare un sistema che fosse portabile – l'uso di librerie esterne era pertanto proibito.
Nonostante questo divieto, fu necessario comunque ricorrere a una libreria esterna gioco-forza, vista l'assenza di soluzioni native. Si prese pertanto la libreria LXML, specificatamente il suo metodo ETREE

Il primo degli script.py, URLreader, leggeva un indirizzo internet, per poi restituire una variabile ElementTree, sotto-tipo del tipo Element

Una parte conveniente taciuta fino adesso: era necessario per Python poter leggere dapprima la pagina internet. A tale scopo mi fu fornita una libreria standard usata dall'Applab in versione ridotta, che implementava una nuova classe, tale ThinBrowser. Fondamentalmente, implementa due funzioni:

- Urlopen: data una url, aggiunge alcuni header per la lettura online, lasciando poi svolgere l'effettivo lavoro di apertura all'omonimo metodo urlopen della libreria nativa urllib (o urllib2 in caso di 2.x)

- gzipPage: data una pagina ottenuta via Urlopen, la trasferisce in un file gzip mediante flusso di bytes, per poi restituirla così "impacchettata".
  - Così processata, può essere trattata in qualsiasi maniera si preferisce. Nel caso in questione, lo scopo è leggerla via il metodo lxml.etree.HTML, trasformandola da pacchetto gzippato a elementTree

3.2.1.1.1.2) Ricerca dell'XPath – Una scelta soggettiva

Stando alla definizione di Python stesso,

The Element type is a flexible container object, designed to store hierarchical data structures in memory. The type can be described as a cross between a list and a dictionary.

Utilizzando pertanto il tipo ElementTree, si genera un dato che permette due funzioni:

- Trattare una stringa come un albero XML, e da lì poi usarlo per generare pagine internet
- Leggere l'albero DOM come un albero XML, e da lì poi navigarlo con la funzione XPath

Ovviamente, l'interesse ricade su quest'ultima funzione.

Una nota interessante: l'XPath non è unico. È possibile raggiungere la stessa informazione in più modi, via via sempre più generici. Questo permette di evitare inutili lungaggini di codice, e in un certo senso fa anche un po' bene per la portabilità. Ovviamente, esiste un punto in cui la "genericità" di un XPath diventa eccessiva, e si iniziano a manifestare degli evidenti "errori" – un po' come quando

aumentiamo le soglie di tolleranza di un filtro passa-banda, e iniziamo ad avvertire sin troppo rumore. Al buon senso del programmatore, tuttavia, decidere quanto essere "generici"

A tale scopo, ripresento un esempio di ricerca XPath per dimostrare meglio queste problematiche:

```
<div class=w_title align=center>
   <table border=0 width=100%>
      <td width=50% class=S18 align=center>Starcitygames Open Series : Dallas / Fort Worth</td>
      <td width=50% class=S16 >#1 Kozilek Caw-Blade –
                <a class=player href=search?player=Gerry+Thompson>Gerry Thompson</a>
      </td></tr>
   </table>
</div>
```

Testo 1 : Porzione di XML con all'interno varie informazioni, tra cui quella del nome dell'evento

L'immagine qui sopra "Testo 1" rappresenta una porzione di pagina HTML proveniente da MTGTOP8. Lo scopo era ricercare l'XPath per poter ottenere l'informazione "nome dell'evento".
Il procedimento mentale è il seguente:

- L'informazione che cerco è: "Starcitygames Open Series : Dallas / Fort Worth";
- Dove si trova? Dentro un td. Ma non un td qualsiasi: è il td con width 50% class 18 align center
- Ma non un qualsiasi td con width 50% class 18 align center. Deve essere anche figlio di una table. con border 0 e width 100%;
- Ma non una qualsiasi table con border 0 e width 100%. Deve essere anche figlia di un div di class w_title e align center;
- Ma non un qualsiasi div class w_title e align center. Deve essere anche figlio di…
- Procedere fino al raggiungimento del nodo Body (o fino al raggiungimento della soglia di sopportazione umana).

Purtroppo l'esempio non presenta dei casi di ambiguità: sebbene l'XPath che personalmente ho scelto è stato:

```
"//table [border=0 width=100]/td [width=50% class=S18 align=center]/text()"
```

In realtà avrei anche potuto fermarmi, considerando solo il pezzo di testo offerto, al solo td width 50 class s18 align center. Una ulteriore analisi della pagina avrebbe poi rivelato che di td class s18 ne esiste solo 1. In tal caso, si sarebbe potuto ridurre il tutto a un magro:

```
"//td[class=S18]/text()"
```

E avere comunque l'informazione desiderata.

Si nota pertanto come sia possibile essere sempre più precisi, ma anche di come esista una soglia dopo di cui è in-necessario esserlo troppo.

3.2.1.2) L'introduzione di JavaScript

Figura 27: Logo di JavaScript

JavaScript [29] fece ingresso in questa pagina solo recentemente. La maggior parte delle richieste, infatti, erano (e sono) gestite via POST di PHP. Ogni pressione di pulsante, di fatto, ricarica la pagina (con tanti inutili trasporti di variabili di sessione), facendo pesare la maggior parte della navigazione dal lato server (PHP è un linguaggio lato server, dopotutto).

Era però chiaro che alcune funzioni potevano essere tolte dal giogo imposto al server, in quanto per lo più erano solo degli effetti grafici che potevano essere sbrigati dal lato client. Una prima applicazione fu il pulsante "mostra/nascondi", che si limitava semplicemente a cambiare il valore display di un div.

Successivamente, con l'impiego sempre più massiccio di JavaScript si riuscì a far ascendere il codice PHP grazie all'uso di Threads (che, comunque, sono semi-nativi, come spiegherò in un'altra sezione). Quello che una volta era un singolo "mattone" che leggeva 90 eventi grazie a JavaScript diventava 90 chiamate da 1 evento, notevolmente più digeribile e meno bloccante.

[29] http://logo-load.com/4202-JavaScript-logo.html

Il passo finale fu utilizzare JavaScript per poter aggiornare il log presente sulla pagina. In un primo momento, ci furono parecchi grattacapi sul fatto che funzionava a fasi alterne. Venni a scoprire, dopo alcune giornate perse, che il problema consisteva nel fatto che JavaScript – o meglio Ajax – teneva di default in cache i risultati delle varie get/post eseguite, indipendentemente se eseguite via $.get (Ajax) o via XMLHttpRequest.open(GET) (JavaScript). Corretto tale behavior, l'aggiornamento del log finalmente funzionava, senza scomodare con uno script PHP il server. Un'ultima idea fu quella di lanciare lo script aggiorna-log ad intervalli regolari come, ad esempio, ogni 2 secondi. Quest'idea verrà poi utilizzata con la gestione degli scripts (alcune sezioni più avanti), ma ho comunque voluto mantenere il pulsante di lettura manuale per ogni evenienza.

In tutto questo, avevo negato la parte grafica, impegnato com'ero a mettere online le funzionalità base. La mia conoscenza maturata mi ha aiutato nel generare, via JavaScript, un sistema di tooltip: al passaggio del mouse sopra un pulsante, un div aggiornava il suo testo, spiegando brevemente la sua funzione. Un'alternativa era quella di generare un mini-popup al passaggio del mouse su pulsante, ma trovai questa soluzione più facile da implementare e, sotto certi versi, anche meno pesante a livello di risorse usate.

3.2.1.3) L'uso di MySQL

Figura 28: Loago di MySql

Tutte le informazioni ottenute via web scraping dovevano essere immagazzinate da qualche parte. La risposta più naturale a questa richiesta era la generazione di una Base di Dati virtuale. E un'altrettanta naturale risposta mi fu data dal committente, che mi propose l'uso di MySQL [30] come struttura dati.

MySQL, assieme a postgreSQL, sono state le due strutture di Basi di Dati insegnatemi, e pertanto anche in quel momento trovai la soluzione conveniente. A differenza, tuttavia, di quanto accadde con Python, non ebbi mai a pentirmi di questa scelta "avventata". La rappresentazione dei dati che avevo scelto era semplice quanto efficace, e al tempo non avevo nemmeno bisogno di cose più complicate quali view e tabelle temporanee. In aggiunta, in ogni mia operazione ero osservato e supportato da un ottimizzatore che, silenziosamente, permetteva di sbrigare in fretta inserimenti e ricerche dandomi addirittura l'illusione che era tutto merito mio e di aver scritto delle query fatte bene.

La connessione a MySQL è sbrigata per la maggiore da Python, anche se verso le ultime parti del progetto cercai un modo di connettermi anche via PHP. Questa volta l'iter fu inverso: in un

[30] https://en.wikipedia.org/wiki/File:MySQL.svg

primo momento mi accontentai di una libreria locale, tale PYMSSQL. Avevo, sinceramente, letto "pymysql" e non avevo pertanto capito che la libreria era pensata per interfacciare python e un server SQL microsoft – utile in locale visto che è il progetto in principio è stato sviluppato su un sistema operativo windows, ben poco utile nella realtà visto che il server è in Debian. Cercai allora meglio, e trovai effettivamente una libreria, di nome PyMySQL, che crea un client semplice python-mysql. Tuttavia, la libreria risultava esterna e, peggio ancora, ancora in sviluppo. Pertanto, l'idea fu scartata.

Impossibilitati però a trovare una soluzione nativa python, ma al contempo impossibilitati anche a scegliere una soluzione esterna arbitraria, si scelse un compromesso: una soluzione esterna scritta in Python. Per nostra fortuna, trovammo MYSQL-CONNECTOR-PYTHON, driver MySQL scritto in Python. Accettare di usare una soluzione esterna portò chiaramente a tutti i vari grattacapi che ne conseguono: quale versione usare? Il server avrebbe potuto supportarla? E se sì, fino a quando?

Ci fu un certo imbarazzo, in principio, quando si scoprì che tale libreria era già stata installata sul server. Una rapida ricerca permise di capire che la versione installata era per il Python 2.x , mentre si stava lavorando in ambiente 3.x (specificatamente, 3.3). La versione del server Debian 7.7 (quindi che supportava fino a Python 3.2, per giunta) rendeva impossibile installare la versione più recente, 2.0.1, in quanto troppo moderna. Al contempo, la versione 1.2.3 sembrava funzionare.

Davanti pertanto alla scelta:

- Regredire alla versione 3.2 di Python e utilizzare la versione 1.2.3 di MySQL-connector

- Aggiornare la VM con Python 3.4 e caricare la versione 2.0.1 di MySQL-connector (e varie ed eventuali altre librerie)

Si scelse quest'ultima. L'unica "scomodità" che la cosa presentava era il dover definire la path assoluta della macchina, invece che semplicemente invocare un comando "Python 3". Questo, infatti, avrebbe richiamato la versione di Python 3 della macchina, che non aveva evidentemente installato il connettore per via del sistema operativo obsoleto.

3.2.2) Lo sviluppo nel tempo

3.2.2.1) La prima versione monotasking

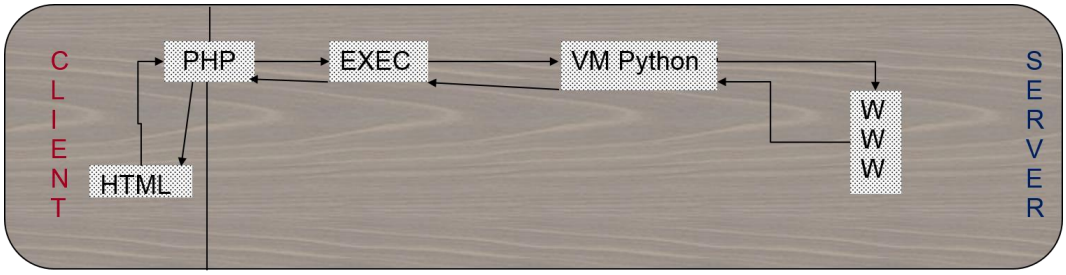


Figura 29: Rappresentazione grafica della prima versione di Webscraper. Il Client dialoga via PHP con il Server, che istrada le richieste alle librerie Python via EXEC

Dopo aver verificato in locale la corretta esecuzione dei vari script Python, era giunto il momento di metterli online e fruibili ai vari altri utenti [31]. Al tempo battevo un terreno totalmente inesplorato, quindi ci accontentammo di fare un ambiente che potesse far partire almeno uno script.

Come già accennato, gli script sono eseguiti via exec. Nello specifico, un exec che indica alla VM ad hoc di Python di leggere un certo file .py , nel seguente costrutto:

[31] Immagine personale – tratta dalla mia presentazione "MTGTop8: netdecking with Python"

```
exec( [path assoluta VM python] [path file .py] [argv1] [argv2] )
```

Era stato stabilito, così, il primo contatto di comunicazione tra PHP e Python in remoto.

Sebbene era possibile in questo modo far partire un singolo script, era possibile passare più argomenti che, nella maggior parte dei casi, andavano ad indicare un range di eventi su cui eseguire il codice. Gli script python furono riscritti in modo da poter accomodare questa necessità.

3.2.2.2) La versione attuale a Thread

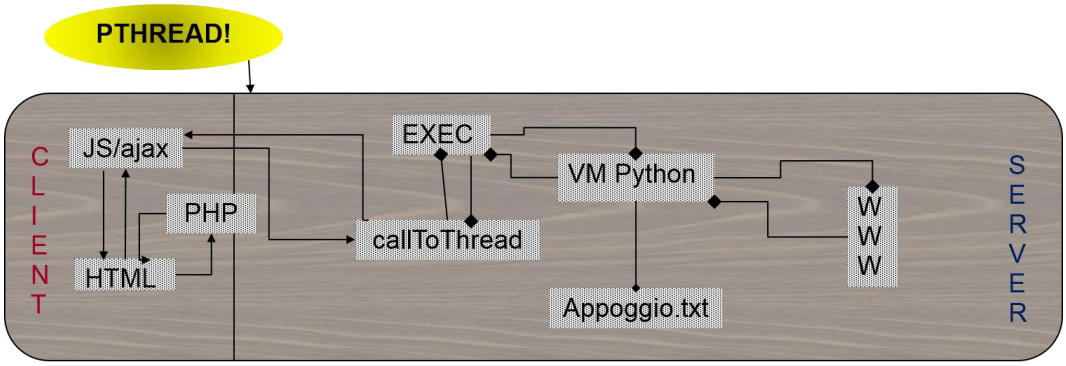


Figura 30: Rappresentazione grafica della versione più recente di Webscraper. Alcune funzioni sono delegate a script Ajax/JavaScript, mentre PHP ora gestisce più chiamate EXEC alle librerie Python

Iniziò però a nascere la necessità di eseguire qualcosa di un po' più complicato e raffinato. Era chiaro che lavorare in monotasking poteva essere una buona partenza, ma era necessario evolversi in un ambiente a thread, anche per altre ragioni che spiegherò successivamente ([32])

[32] Immagine personale – tratta dalla mia presentazione "MTGTop8: netdecking with Python"

I Thread in PHP sono un argomento abbastanza "tricky". Infatti, il server dedica un singolo thread al PHP – nel senso: anche se si viene a instaurare un ambiente multithread in PHP, il server comunque lascia a disposizione degli stessi una quantità di spazio limitata – per giunta, qualora uno di questi script causasse un crash, tutti gli altri vanno automaticamente in fallimento. Esiste una soluzione nativa di PHP, consistente in nulla più di una opzione del file PHP.ini. Si tratta del valore "Thread Safety", che permette l'accesso alla libreria PThread. Per via di alcune ragioni legate alla sicurezza, di default tale valore però risulta essere a disabled, causando un errore "Class Thread not Found" e, di fatto, impedendo a un ambiente multitasking di essere creato.

Una volta apportate queste necessarie modifiche all'ambiente PHP, si rivelò anche necessario togliere il peso di eseguire la exec dalla pagina principale, e relegare tale compito a un altro file PHP. Per fare questo, si è ricorso a JavaScript/Ajax, che con il suo "Asynchronous" permetteva di inviare più "pacchetti" di scripts insieme. Si parla di pacchetti perché, di fatto, questo veniva eseguito ogni volta che uno script che tratta un range di eventi veniva invocato.

Mentre una volta c'era una singola exec che trattava N argomenti, ora c'erano N exec che trattavano 1 argomento. Una parallelizzazione che non solo sveltiva il lavoro (di fatto, il risultato della lettura dell'evento N era ininfluente per quello dell'evento N+1 – al più ci sarebbero stati alcuni disordini negli ID interni del database, ma nulla di grave), ma permetteva anche di avere dei risultati intermedi durante l'esecuzione. La pagina principale era infatti "sbloccata" e dopo il piccolo tempo necessario ad impacchettamento e spedizione era possibile di nuovo fruire dei suoi servizi mentre si aspettavano i vari ritorni.

3.2.3) Problemi incontrati (e risolti)

3.2.3.1) EXEC e il suo potere bloccante

Si è convenientemente taciuta finora la ragione scatenante della necessità del passaggio a Threads. Non era infatti solo una questione di avanzamento tecnologico, o di rapidità di esecuzione. Era anche e soprattutto per un problema intrinseco della funzione EXEC

EXEC, infatti, è bloccante. Una volta eseguitala, il server PHP si congela, impegnato com'è nell'esecuzione dello script, e si scongela solo una volta terminata l'esecuzione del comando. Questo era:

- Altamente spiacevole, perché di fatto bloccava la navigazione;
- Assolutamente fuori questione, perché uno script troppo lungo rischiava di far scattare la protezione da timeout di circa 30 secondi del server, di fatto non permettendo tali funzioni in casi di grande mole di dati da processare. E non si parla solo di un cluster di eventi – a volte anche un singolo evento troppo grande rischiava di diventare illeggibile così.

Si voleva da subito tentare di eliminare questa capacità bloccante, ma il pensiero non andò subito ai thread. Si voleva, anzi, trovare quale behavior di EXEC dava il potere di bloccare, ed eliminarlo. E le ricerche portarono, in effetti, a un frutto.

Scoprii che Exec era bloccante fintanto che non veniva dichiarato dove ridirezionare lo STDOUT. In caso infatti di una locazione indicata come "dump", il PHP poteva continuare l'esecuzione una volta fatto partire lo script.

Le prime ricerche suggerivano di riscrivere l'allora script:

```
exec( [path assoluta VM python] [path file .py] [argv1] [argv2] )
```

in:

```
exec( [path assoluta VM python] [path file .py] [argv1] [argv2] > dev/null )
```

Come ebbi modo di intendere più tardi, questa soluzione permetteva l'esecuzione di script "stealth". In sintesi, stavo dicendo al server, "eseguimi questo script python, poi prendi lo stdout e buttalo via".

Non era ovviamente la soluzione che volevo. Se mi ero preso la briga di mettere delle print nei miei script era per dare allo stesso un tentativo di dire il suo ACK durante l'esecuzione, o alla peggio di capire dov'è che si era bloccato.

Mi era chiaro, però, come ridirezionare uno stdout, ovvero con il solo uso di ">". A quel punto, era necessario pensare a un'altra locazione. Dove poter mandare una moltitudine di stringhe returnate? La risposta più ovvia fu, "in un file txt".

E così la nuova versione, attuale, è:

```
exec( [path assoluta VM python] [path file .py] [argv1] [argv2] > /appoggio.txt)
```

in questo modo, il risultato era tutto mandato a un file di testo locale, che poi veniva letto a fine esecuzione.

3.2.3.2) Python e UTF-8

Un grosso grattacapo contro cui mi scontrai fu un errore di comunicazione html-python-PHP per quanto riguardava il formato da usare.

Difatti, nei casi in cui veniva letto il nome di qualche evento straniero (come il temibile "Nationals Nouvelle-Zéland", che fu scelto come soggetto-cavia per testare le varie soluzioni), o peggio ancora l'arrivo di giocatori stranieri, (tra cui cito Cristian Zúñiga, Leon Lindbäck , Olle Råde, Bertrand Lestrée , Øyvind Andersen), il PHP iniziava a lamentare problemi di comunicazione, dichiarando:

```
'ascii' codec can't encode characters in position [posizione]: ordinal not in range(128)
```

A quel punto, dovetti ricontrollare tutto il processo di lettura, e capire dove stava il problema di traduzione. Analizzai i vari punti di possibile "diverbio":

- Mtgtop8.com stesso
- Webscraper stesso
- PHP stesso
- Passaggio argv Webscraper -> Python
- Parsing Mtgtop8 -> Python
- Python -> Python (eventuali riconversioni nascoste, magari per casting a stringa di variabili indefinite)
- Python -> Webscraper.php
- Python -> mySQL
- mySQL stesso

Potei istantaneamente cancellare gli ultimi due punti: era infatti interessante notare come MySQL non stava dando assolutamente problemi. Per il database, gli stava arrivando uno “Zélande”, e “Zélande” registrava.

Iniziai pertanto una caccia per tutti gli altri punti:

- Mtgtop8.com era il problema?
  - NO?
  - Invero, scoprii che la codifica del testo di mtgtop8.com era in windows-1252, che scoprii essere un ASCII. Fu il mio primo indiziato per parecchio tempo, e pertanto mi ero assicurato che in Python venisse utilizzata la codifica UTF-8 corretta. Ma qualcosa non mi tornava: se era davvero un formato Ascii, perché sulla pagina i caratteri erano mostrati correttamente?
- Webscraper era il problema?
  - NO!
  - Avevo cura di salvare il file .PHP con formato UTF-8 e non ANSI, avevo inserito alcuni header che obbligavano la pagina ad essere in formato UTF-8, forzavo l’UTF-8 anche all’interno della textarea che era il log. Avevo preso ogni possibile precauzione che mi poteva venire in mente sul problema, pertanto esclusi con ragionevole certezza questa cosa.
- PHP stesso era il problema?
  - NO
  - Chiesi al sistemista di forzare, nel PHP.ini, l’uso di UTF-8 come standard. Probabilmente era già settato in quella maniera, ma non potevo lasciare adito a qualsiasi dubbio.
- L’argv di Webscraper verso Python era il problema?

- NO
- Semmai, era una soluzione. Quando Python prende degli argomenti, può imporre un controllo di error handling, "surrogateexcept". Questo impone il behavior da rispettare qualora si ritrovasse un problema di caratteri non-ASCII. Conclusi rapidamente che potevo in casi di emergenza rivolgermi a lui se il problema non fosse stato risolto. In aggiunta, mi assicurai che l'argv non fosse passato via unicode.

- Il parsing MTGTOP8.com -> Python era il problema?
  - NO
  - Anche se scritto sotto variabile ElementTree, la pagina era letta correttamente e salvata come UTF-8, poiché scoprii che tale codifica era lo standard dai tempi di Python 3.x. In aggiunta, lxml è una libreria che lavora in unicode, e anche thinbrowser restituiva la pagina come stream re-interpretato poi anche esso in unicode da Python
- Python cambiava codifica dentro di sé?
  - NO
  - Come detto, la codifica standard di una variabile stringa era utf-8. A quel punto, bastò fare un cast str di ogni variabile prima della print, assicurandomi pertanto l'effettivo formato UTF-8 dell'output
- Il passaggio dati Python -> Webscraper era il problema?
  - BECCATO!

Fu con qual certo sbigottimento che scoprii che mentre, invero, lo STDIN e lo standard di ogni stringa era in UTF-8, lo STDOUT veniva ricodificato in ANSI_X3.4-1968. Ovvero, un ASCII. In sintesi, potevo dare come argomento delle print una stringa in qualsiasi

formato – Python l'avrebbe comunque esportata in formato ASCII, e da qui poi l'errore "**ascii** codec cannot…"

Il mio committente, però, mi propose una sfida: riuscire a cambiare la codifica senza andare a toccarla "fisicamente" come variabile. Un dilemma, a prima vista, che somigliava a quello di avere nipoti ma di non diventare nonni.

In un primo momento, mi fu suggerito di cambiare il codec da remoto. In effetti, la "formula magica" era semplice:

```
sys.stdout = codecs.getwriter('utf8')(sys.stdout)
```

Come lo feci, però, veniva generato un errore che non riusciva ad essere contenuto nemmeno dal try-except più generico che potevo immaginare. Dovetti pertanto scartare la soluzione.

Altra soluzione poteva essere lavorare con il surrogateexcept e risolvere il problema alla radice, alla lettura di mtgtop8. Una possibilità consisteva nell'usare "ignore" come parametro. Ovvero, "quando incontri un carattere non-ascii, ignoralo completamente". Ciò significava leggere "Zélande" come "Zlande". Storsi molto il naso, e diressi lo sguardo da altre parti. Perché, infatti, mortificare mtgtop8 quando la colpa era solamente di Python?

Altri miei colleghi di studio allora suggerirono l'uso di una libreria pypi, tale UNIDECODE, che permetteva di cambiare i caratteri non-ascii nel carattere "normale" più vicino. Ad esempio, una "è", o una "è" venivano cambiate in "e". Stavolta fu il committente a storcere il naso, ribadendo che non voleva utilizzare dei software esterni – in più, una lettera cambiata in qualcos'altro non era

troppo diverso dall'ignorarla completamente, come faceva la soluzione precedente.

Di fronte a queste impossibilità di procedere, mi fermai e iniziai a ragionare su un compromesso. Trassi alcune conclusioni:

- La comunicazione python->mySQL era corretta. E, in effetti, per lo scopo della pagina, mi bastava questo. L'errore era di mera visualizzazione del PHP
- Le print che eseguivo erano per sincerarsi, fondamentalmente, del corretto funzionamento dello script. Avrei potuto rimpiazzarle tutte con un "va bene!", e lo scopo non sarebbe stato intaccato minimamente
- Il log che andavo a scrivere era completamente volatile. Non sarebbe mai stato salvato, e pertanto non avrei mai avuto il problema di un programma che sarebbe andato a leggere quei dati, li avrebbe trovati incomprensibili e avrebbe crashato

Fu a quel punto che utilizzai una funzione delle stringhe di python, ovvero encode. Indicando "utf-8", avrei forzato la stringa a rimpiazzare ogni carattere non-ISO con il relativo UTF-8 Literal e, quel che è meglio, di printarlo codificato come se fosse un flusso di bytes. "Zélande" sarebbe stata stampata a video come "b'z\xc3\xa9lande'".

Una soluzione grezza, ma che accomodava le mie richieste.

3.2.3.3) Controllare gli Scripts

Uno dei problemi più grandi derivati dall'uso di script esterni riguardava la capacità di controllarli – non c'erano infatti delle

soluzioni native e lampanti, a livello PHP/JavaScript per poter gestire questi script, .py, che:

- Erano generati da una VM Python ad hoc…
- …A sua volta chiamata da una Exec di PHP…
- …che veniva chiamata da un Ajax.

Nella versione monotasking, come soluzione utilizzati quello che battezzai il "flag di terminazione". In sintesi:

- Lo script veniva fatto partire;
- Alla pressione di un pulsante "fermati", veniva generato un file txt dal nome fisso;
- Gli script, al loro interno, facevano dei controlli a ogni punto critico, ovvero qualsiasi operazione che poteva richiedere molto tempo (quindi, per esempio, a ogni lettura di pagina internet, oppure prima di inviare una query), e verificavano se tale file txt esisteva;
    - se esisteva, lo cancellavano, e poi procedevano ad un abort "grazioso";

L'idea funzionò fintanto che si era in modalità monotasking. Una volta che i Thread furono implementati, si rischiava che un processo poteva leggere per primo l'evento, cancellarlo e terminarsi, mentre tutti gli altri sarebbero andati avanti ad oltranza. Pensai allora a una sorta di reverse-flag, ovvero:

- Quando lo script partiva, generava un file txt all'interno di una cartella, come nome del file si usava il proprio PID, generato durante l'esecuzione python;

- gli script controllano, periodicamente, se tale file ancora esiste;

- alla pressione di un pulsante “fermati”, tutti i file .txt all'interno della cartella vengono cancellati;
  - Se lo script notava l'assenza del proprio file.txt procedeva ad un abort “grazioso”;
- se lo script terminava correttamente, procedeva a cancellare il proprio file txt;

Ma, a questo punto, mi scontrai con due problemi:

- mancanza di permessi : apparentemente, avevo il permesso “777” su una cartella, ma non sulle sue sottocartelle: potevo creare e rinominare, ma non cancellare via PHP;
- apparente impossibilità di eseguire il comando Kill da PHP, come da impostazioni del server (“ovviamente non lo permettiamo”).

Per lungo tempo, pertanto, gli script così eseguiti potevano avanzare fino a terminazione naturale, impossibilitato a trovare una via d’uscita da questo dilemma. Questo status quo durò fino al consiglio offertomi da un ragazzo dell’Applab, che si era ritrovato nello stesso tipo di problema e aveva trovato una soluzione quantomeno ingegnosa e a cui, ammetto, non sarei arrivato nemmeno dopo mesi di ponderazione: appoggiarsi su MySQL.

La sua tecnica – che poi adattai alle mie necessità – è la seguente:

- Quando uno script viene fatto partire, questo contatta il DB, scrivendo in una tabella ad hoc alcune informazioni, tra cui il proprio PID, recuperato via funzione `os.getpid();`
  - Come feature aggiuntiva, è in grado anche di calcolare un valore che indica, all’incirca, il proprio completamento.

Prevede quanti "lavori" deve fare, e ogni volta che ne conclude uno tiene traccia di questo progresso;

- Quando il completamento raggiunge il 100%, oppure viene richiesto tramite apposito pulsante, il processo manda il proprio PID a 0 (Questo nella versione originale di Alessio. Nel mio caso, non viene fatto niente);

- Nel mentre, sulla pagina internet è sempre presente una funzione che interroga il DB a intervalli regolari. L'intervallo è regolabile, per ora, solo da riga di codice
  - Ad ogni intervallo, la pagina aggiorna un div, che si vede riscritta una tabella di "lavori in corso". La tabella mostra il nome del processo, la % di completamento e dispone del pulsante di arresto sopra accennato. Vengono letti solo i processi il cui status non è a 0 (ossia sono terminati)
  - Nel momento in cui viene letto un processo con completamento al 100% (nel senso che il contatore di quel processo è uguale al numero di compiti da fare), la riga di quel processo viene scritta con uno sfondo verde, e il pulsante di arresto viene disabilitato. In aggiunta, viene eseguito un update, che manda lo status di quel processo a 0. Infine, manda anche una richiesta di lettura del log, che ora avrà sicuramente nuove informazioni al suo interno
    - Combinato con il filtro allo status, questo significa che un processo terminato rimarrà su schermo per il tempo di un intervallo, prima che esso non venga più letto

Questo approccio è tuttora presente e funzionante, anche se comporta alcune problematiche minori:

- Deve comunque essere possibile per il demone PHP eseguire dei comandi KILL, oppure i pulsanti di arresto manuale continueranno a non funzionare;
- La tabella dei processi, di fatto, diventa uno "storico", poiché nessuna riga al suo interno viene mai cancellata – viene solo mandato a 0 un certo valore. Nel codice è comunque presente una funzione di "pulizia" dei processi terminati, anche se non c'è al momento nulla che la richiama, visto che la funzione di "storico" potrebbe essere di interesse generale. Attenzione, però, a non far crescere le dimensioni della tabella in modo eccessivo;
- Ad ogni intervallo fisso (settato, per il momento, a 2 secondi), viene lanciata una richiesta a MySQL per l'interrogazione. Più, alla fine di ogni lavoro viene lanciato un update. La costanza dell'interrogazione, e la potenziale rapidità degli update (è possibile per degli eventi di sola lettura terminare in ancora meno di 2 secondi), c'è un significativo overhead verso MySQL.

3.3) Configuratore.PHP

Figura 31: Schermata di Configuratore.php

Sbrigata la pratica del web scraping, e caricato il database di sufficienti informazioni (13.000 eventi in circa 2-3 giorni di upload martellanti a 50 eventi a "pacchetto"), era giunto il momento di provvedere all'utility di creazione mazzi. Tuttavia, si voleva cercare un approccio più statistico e meno arbitrario sulla costruzione di un mazzo – era per questo, dopotutto, che si era fatto netdecking. La richiesta del committente a tal proposito era: "*generare un utility di creazione mazzi: scegliendo tra una topX di carte più "famose", si continua poi a costruire il mazzo usando come "suggerimento" le carte più giocate insieme ad una certa combinazione di carte – l'intero mazzo per default. Libertà di scelta sui mezzi da usare*". ([33])

([33]) http://mtgweb.dmi.unipg.it/mazziapp/magicacademia/core.php

Cerco di rendere tale richiesta più comprensibile:

Figura 32: La carta "Flash"

Supponiamo di voler prendere la carta "Flash" [34].
Basandosi sulle combinazioni ricercate, ci si può aspettare che tra i suggerimenti ci sia una di queste carte:

- Protean Hulk. Questo perché esiste un tipo di mazzo, chiamato "Flash Hulk" che si basa esattamente su questa combinazione: Flash+Protean Hulk;
- Black Lotus [35]. Questo perché è la carta più giocata nei formati Vintage e Legacy, quindi è altamente probabile che "involontariamente" questa carta sia stata giocata assieme a Flash;
- Island: Questo sia perché Flash è una magia blu (quindi è ragionevole aspettarsi che la terra che genera mana blu sia giocata assieme a una magia blu), sia perché Island è la carta più giocata di Magic (circa il 40% dei mazzi di Magic gioca una Island), finendoci pertanto "involontariamente" tanto quanto black lotus

[34] http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=3337
[35] http://www.geeksdiet.com/blog/pay-100000-playing-card/

Considerato che la maggior parte degli eventi è in Standard, probabilmente vincerà Island sopra Black Lotus.

(Alcuni test eseguiti per pura curiosità hanno poi confermato questa ipotesi: apparentemente Island è la 3° carta più famosa, al pari di Demonic Tutor, carta usata quasi quanto Black Lotus. La carta più giocata assieme a Flash, in assoluto, è Polluted Delta, giocata in 94 dei 100 mazzi che usano Flash. Black Lotus non rientra nemmeno nella top25 di carte più usate in combinazione con Flash).

3.3.1) Focus: staple e semi-staple – minaccia alle statistiche

Figura 33: La carta "Back Lotus", la più famosa (e giocata) carta di Magic: The Gathering

Qualcuno potrebbe argomentare che “protean hulk” è quella più indicata ad essere suggerita come “compagna” di Flash. In effetti, è l’unica che:

- Non fa parte di quel gruppo di carte prevedibili che vengono giocate praticamente ovunque (semi-staple, in gergo);

- Non è una carta che puoi mettere in un qualsiasi mazzo e aspettarti di usarla senza complicazioni (staple);
- In genere, viene giocata solo ed esclusivamente abbinata a Flash, poiché è una delle poche carte che la rende competitiva in ambiente professionale.

Purtroppo, chi fa questa argomentazione sa anche che la definizione di "staple" e soprattutto di "semi-staple" è molto soggettiva, soprattutto in Magic. Un tempo, la carta più giocata in ambiente Standard era "Mutavault". Ora quella carta è uscita fuori da quel formato, ed è stata rimpiazzata da "Smuggler's Copter". Mantenere, pertanto, una lista di carte da considerare "semi-staple" o "staple", affinchè esse vengano filtrate fuori dai risultati per favorire le "giuste carte" da inserire è:

- Non-automatizzabile (richiederebbe comunque qualcuno che effettivamente gioca che può dire quali sono tali carte);
  - Invero, MTGTOP8 mantiene una lista della top10 di carte più usate in ogni formato. Ma è giusto definire "semi-staple" Smuggler's Copter, giocata dal 56% dei mazzi, tanto quanto Declaration in Stone, giocata dal 25% dei mazzi? Oltre quale soglia % una carta è definibile come semi-staple?
- Non desiderato, in quanto in questo modo sarebbe impossibile inserire alcune carte, tra cui soprattutto le terre base, all'interno del mazzo.

3.3.2) Scelte di Sviluppo

3.3.2.1) La Consapevolezza di PHP e JavaScript

La nascita di Configuratore.PHP è relativamente recente, e avevo finalmente abbastanza consapevolezza di potenzialità, errori comuni e soluzioni incontrabili via PHP e Ajax/JScript. Decisi pertanto di sviluppare tutto completamente in ambiente PHP-JScript, ritenendolo molto più digeribile e meno complicato di un eventuale collegamento ad altri linguaggi di programmazione quali script Java o script Python richiamati via exec.

Avevo già sperimentato ai tempi di Webscraper un metodo per poter permettere al PHP di connettersi a MySQL. La libreria, al tempo, era "mysql", ma dopo un anno circa ero passato oltre il periodo di transizione, in cui questa libreria veniva deprecata e , alternativamente, si consigliava sempre la nativa "mysqli".

Risolto il problema più grande, dovevo solo trovare un modo per poter lavorare con variabili di sessione anche in ambiente JavaScript. Così come PHP utilizza un array, $_SESSION, per le variabili di sessione, JavaScript usava ben due di questi array:

- localStorage, che mantiene una variabile di sessione anche dopo la chiusura della pagina, e che viene resettata solo al successivo riavvio del browser web:
- sessionStorage, che mantiene una variabile di sessione fino alla chiusura della pagina.

In un primo momento andai di localStorage, per poi correggermi con sessionStorage - non avevo davvero bisogno di dati così permanenti… Dovevo però sottostare a una limitazione: dentro questi array potevo inserire solo ed esclusivamente stringhe.

Quindi qualsiasi dato non-stringa (inclusi array di stringhe) sarebbe stati convertiti in una stringa via JSON. JSON.stringify e JSON.parse avrebbero permesso la scrittura e la lettura di tali variabili rispettivamente.

La parte più delicata fu eseguire script JavaScript e PHP insieme. La cosa è "tecnicamente impossibile", poiché JavaScript è lato client e PHP è lato server. Riuscii comunque a stabilire una comunicazione tra i due linguaggi in questo modo:

- in caso di JavaScript-to-PHP, scrivendolo direttamente sulla pagina, nei costrutti:
  - `<script>print "<?PHP [comando PHP da eseguire] ?>";</script>`
    - per eseguire uno script generico
  - `<script> document.getElementBy…[selettore].innerHTML= "<?PHP [comando PHP da eseguire] ?>"; </script>`
    - per aggiornare il contenuto di un DOM in modo "dinamico";

- in caso di PHP-to-JavaScript, bastava scrivere qualcosa del tipo:
  - `echo "<script> [comando JavaScript] </script>";`

In ambo i casi, le modifiche erano molto piccole –ad esempio, JavaScript non avrebbe mai potuto richiamare funzioni PHP – e a volte andavano anche contro la natura stessa dei linguaggi (ad esempio, le chiamate a JavaScript via PHP sarebbero state eseguite solo come ultima azione, in quanto avvenivano dopo il caricamento della pagina).

In tutto ciò, iniziai a prendere la buona abitudine di mettere gli script di JavaScript in fondo alla pagina. Più di un problema di "manca un elemento DOM che abbia un ID di nome [nome]" fu risolto semplicemente spostando un po' più sotto o un po' più sopra lo script problematico…

3.3.3) Problemi incontrati (e risolti)

3.3.3.1) Cronache di una query – come MySQL può uccidere un demone PHP

Figura 34: Allegoria dell'uccisione del demone PHP da parte di MySql in un fotomontaggio da me effettuato sulla carta "*Akroma's Vengeance*"

Probabilmente, questa è stata la parte più saliente del mio intero progetto di tesi – l'ottimizzazione della query di ricerca delle carte da suggerire [36].

L'ottimizzazione nasceva non dalla semplice richiesta di prestazioni migliori (limitate, comunque, dal fatto che l'oggetto di questa query

[36] Immagine personale fotomontaggio

è una mastodontica tabella da 2.2 milioni di righe in costante crescita), quanto piuttosto un bizzarro comportamento che uccideva – e uccide tuttora – il demone PHP del server.

È necessario trattare questa query a "gradini", iniziando dal più basso: come trovare la carta più "famosa" in assoluto, ovvero la più giocata.

Una carta ha un grado di fama che dipende da quanti mazzi la giocano e, pertanto, il suo ID dovrebbe comparire molte volte quando lo si ricerca nella tabella in questione. In tal caso, basta eseguire una query semplice del tipo:

```
SELECT cardID FROM decklists GROUP BY cardID ORDER BY COUNT(*)
DESC
```

Ossia:

- SELECT cardID (restituisce l'ID delle varie carte)
- FROM Decklists (la tabella interessata)
- GROUP BY cardID (necessario poiché, altrimenti, COUNT(*) conterebbe solo quanti valori cardID diversi ci sono nella tabella
- ORDER BY COUNT(*) (ordinati secondo il loro valore count…)
- DESC (…affinchè quello con il count più alto stia in cima)

Ora, si spiega come cercare la carta più "famosa" giocata assieme a una specifica.
Si suppongano queste decklist parziali, in formato {nomemazzo: [ID,…,ID]:

Mazzo ALPHA : [2,5,8,9,22,35]
Mazzo BRAVO : [2,5,7,8,20, 59]
Mazzo CHARLIE : [4,5,10,30,50]
Mazzo DELTA : [2,5,8,9,10,56]
Mazzo ECHO : [50,51,54,55,56]

Si vuole sapere quale è la carta giocata in combinazione di più con la carta che risponde all'ID 2.

Umanamente parlando, faremo così:

- Prendiamo ogni mazzo, vediamo se dentro di esso c'è la carta 2;
  - Se c'è, lo terremo da parte;
  - Se non c'è, scarteremo l'intero mazzo;
- Abbiamo solo alcuni mazzi da parte. Tutti questi mazzi hanno dentro di sé la carta 2;
- Prendiamo le carte dentro questi mazzi, e contiamole;
  - Ciò includerà anche la stessa carta 2 come giocata più in combinazione con la carta 2. Questo behavior è stato corretto, ma per il momento trascuriamolo;
  - La ricerca è pseudo-distinct:
    - Se è la prima volta che incontriamo una carta nuova, la teniamo a mente;
    - Se non è la prima volta, consideriamo quella carta con un "+1";

Nell'esempio, pertanto, accade questo:

- Esclusione del mazzo Charlie ed Echo, in quanto non hanno la carta 2. Rimangono 3 mazzi;
- Guardiamo ogni mazzo e contiamo le varie carte:
  - Risultato (formato: {ID:Count()}

- {2:3 | 5:3 | 7: 1| 8:3 | 9:2 | 10 : 1| 20:1 | 22:1 | 35 : 1 | 56:1 | 59:1}

- Riordinando il tutto in termini percentuali (basato su 3, il numero di mazzi totali):
  - Formato: {%:[ID,…,ID]}
  - {100%: [2,5,8]}{67%:[9]}{33%:[7,10,20,22,35,56,59]}

Quindi ci aspettiamo che tra le carte suggerite ci siano prima 5 e 8, poi 9, poi tutte le altre. L'ordine "interno" è indifferente.

Implementiamo pertanto la cosa in termini mySQL:

- Query: seleziona tutti i mazzi con la carta 2;

```
SELECT ID_Mazzo FROM Decklists WHERE Decklists.cardID='2'
```

  - Costo: O(n) , ove "n" è il costo della lettura di Decklists;

- Query: conta le carte in quei mazzi:

```
SELECT cardID, COUNT(*) FROM Decklists WHERE ID_MAZZO IN{ query qui sopra} GROUP BY cardID ORDER BY COUNT(*) DESC
```

  - Costo: O(n) + O(n) = O(2n), poicéè Decklists è letta due volte (una volta dalla query interna, una dalla query esterna);

Questo è il costo minimo che una query di ricerca di carta famosa può avere. O, quantomeno, se esiste un metodo che richiede 1 sola lettura di Decklists invece che 2 minimo, non l'ho ancora trovato.

Ora, la ricerca della carta più usata in combinazione con una coppia (o comunque sia una tupla) di carte è già più interessante come procedimento. Un approccio "iterativo" potrebbe essere del tipo:

- Trova tutti i mazzi che hanno la carta X;

- Trova tutti i mazzi che hanno la carta Y e che compaiono nell'elenco di quelli che hanno la carta X;
- Trova tutti i mazzi che hanno la carta Z e che compaiono nell'elenco di quelli che hanno la carta Y (e di conseguenza anche la X);
- …
- Decklist dei mazzi trovati dall'ultima iterazione;

Come si può notare, questo approccio costa un assurdo (On + mn), ove m è il numero di carte che si stanno cercando. Considerate le dimensioni della tabella, questo approccio era assolutamente improponibile.

Dopo aver ricevuto consiglio su chi chiedere per ottimizzare una query, e dopo aver ricevuto l' "illuminazione" da uno dei professori della facoltà in modo quasi accidentale, cambiai il tipo di approccio.
Quando affrontai il problema la prima volta, ovvero cercare le carte che sono in combinazione con una tupla di carte, ebbi un approccio semantico: stavo cercando le carte dei mazzi che avevano la carta X **e** la carta Y – pertanto scrissi istintivamente: "cardid=X AND cardid=Y". Il risultato, con un qual certo stupore, fu vedermi questo AND riconvertito in un OR, e quindi vedermi ritornati tutti i mazzi che giocavano la carta X **OPPURE** la carta Y, ma non necessariamente ambedue insieme.
Una cosa che notai, però, quando riconvertii l'AND in un OR, fu che se un mazzo aveva ambedue le carte che cercavo, il suo ID compariva due volte. Con una qual certa sicurezza, notai che la cosa era iterativa: se un mazzo ha tutte le N carte che cerco, il suo ID compare N volte.

Apparve pertanto chiaro che potevo leggere la tabella Decklists una singola volta, e ricercare tutti i mazzi, in maniera non-distinct, che avevano almeno una delle N carte cercate. A quel punto, i mazzi interessati erano quelli il cui count era uguale a N.

In termini di MySQL:

```
SELECT cardID, COUNT(*) FROM Decklists WHERE ID_MAZZO IN(SELECT ID_Mazzo FROM Decklists WHERE cardID=[id] OR… OR cardID=[id] GROUP BY ID_MAZZO HAVING COUNT(*) = [m]) GROUP BY cardID ORDER BY COUNT(*) DESC
```

Ossia:

- SELECT cardID, Count(*) FROM DECKLISTS WHERE ID_MAZZO IN: già spiegato nel primo esempio;
- SELECT ID_MAZZO: dammi tutti gli ID dei mazzi…
- FROM Decklists: …della tabella decklists…
- WHERE cardID=[id] OR… OR cardID=[id]: … che hanno almeno una delle carte (id) cercate;
- GROUP BY cardID: contando i diversi cardID (che sono necessariamente nel range degli id ricercati);
- HAVING COUNT(*) = M: e di questi, dammi solo quelli il cui ID compare M volte, dove M è il numero degli ID cercati;
- GROUP BY cardID ORDER BY COUNT(*) DESC: già spiegato nel primo esempio;

Il costo totale, a questo punto, era un O(2n). Non importa di che dimensione era la tupla, la tabella sarebbe stata letta esattamente 2 volte.

Ora, le due query ("trovami i mazzi che hanno tutte le carte" e "dammi la decklist di questi mazzi") sono due query veloci – mySQL impiega una frazione di secondo per eseguirle individualmente. Per qualche ragione che trovo ancora assurda, se combinate causavano uno dei seguenti casi, a seconda di come gli veniva data la query in pasto:

- Via Adminer (tool di controllo remoto online sviluppato in PHP e che anche esso funziona usando mysqli): **l'intero demone PHP del server veniva bloccato**, ed era necessario per me tornare al laboratorio per richiedere manualmente un riavvio di tutte e tre le sue versioni al momento presenti sul server;
- Via linea di comando in locale: ben 16 minuti per essere portata a termine

La soluzione subito successiva fu quella di impiegare una View, poi evoluta in tabella temporanea (per poter comunque sfruttare le capacità ottimizzatrici degli indici, che una view non può avere) che tenesse all'interno il risultato della query interna, di ricerca dei mazzi. L'iter diventò, pertanto:

- Crea una tabella temporanea temp, formata da un campo ID univoco e indicizzato, e un campo ID_Mazzo;
- Query interna: cerca tutti i mazzi che hanno la combinazione di carte richiesta. I vari ID vengono inseriti via INSERT dentro la tabella temporanea;
- Query esterna: rintraccia la decklist di tutti i mazzi il cui ID compare all'interno della tabella temporanea;
- DROP della tabella temporanea, per assicurarsi che sia ripulita alla prossima iterazione;

Questo portò i tempi di esecuzioni in, al più, 15 secondi – che è il tempo che viene richiesto se si cercano le carte in combinazione con la carta più "famosa", Island. A volte questi tempi si dilatano, com'è logico che sia, a seconda di quanto il server del Database è in quel momento impegnato.

Sempre meglio di un freeze PHP e un demone ucciso…

3.3.3.2) Il Carosello fai-da-te

Era giunta l'ora di pensare un po' alla parte grafica. Il mio committente era rimasto affascinato da alcuni "carousel" online, e voleva una simile implementazione anche in questa pagina.

Come "carousel" si intende una sequenza di immagini, simili a una galleria, con l'importante feature di essere cicliche – una volta arrivati all'ultima immagine, si riparte dalla prima in maniera quasi impercettibile. I caroselli, fedeli all'originale, in genere mostrano una singola immagine per volta, ma era invero possibile generare caroselli che mostravano più immagini contemporaneamente. In egual modo, un carosello poteva "scorrere" le immagini singolarmente, oppure a blocchi (cioè, se il carosello aveva dimensione 4, un singolo scorrimento avrebbe caricato altre 4 nuove immagini invece che solo 1 nuova). I caroselli possono anche avere delle brevi didascalie di descrizione, di tali immagini.

Un piccolo problema riguardava il fatto che tutti i caroselli visionati dal mio committente erano soluzioni offerte da programmatori esterni che avevano dedicato un po' del loro tempo libero a css e/o JavaScript. Era necessario per me generare una versione mia, ad-

hoc, che potesse accomodare la richiesta di un carosello di carte, piuttosto che di immagini arbitrarie.

Una soluzione mi fu offerta, in un primo momento, dalla stessa W3C – se decidevo di sottostare al loro css, potevo impiegare un carosello "standard", tale "W3-Carousel". Il css aveva anche lo spiacevole effetto collaterale di modificare lo stile grafico dell'intera pagina, pertanto avrei dovuto personalizzarlo con definizioni mie a cascata.

Il carosello proposto era, invero, quel che cercavo: una galleria di immagini, scorribili singolarmente, ciclica e che disponeva di didascalie. Purtroppo, queste didascalie erano posizionate sulla carta stessa, e questo era un problema. Non c'era infatti un angolo libero dove poter mettere questa descrizione senza soffrire di oscuramenti alle informazioni della carta:

- In alto a sinistra: oscuramento del nome della carta
- In alto a destra: oscuramento del costo di mana
- In basso a sinistra: oscuramento del nome dell'artista
- In basso a destra: oscuramento della rarità
- Al centro della carta: oscuramento dell'immagine, e comunque era un pugno nell'occhio

L'idea però di avere una didascalia personale per ogni carta mi era importate: il box di didascalia poteva all'occorrenza avere al suo interno altre informazioni come, ad esempio, la % di corrispondenza della carta rispetto alla combinazione.

La soluzione giunse rapidamente, dopo un po' di lavori di posizionamento: la didascalia si sarebbe trovata sotto la carta, allungando il div dell'immagine.

Ben più problematico era invece l'implementazione di tale carosello nella pagina. Il carosello, di per sé, è una componente statica, che dovrebbe avere la sua galleria immagini personale già pronta e definitiva. Ma, chiaramente, le informazioni che il carosello doveva mostrare erano invece dinamiche o, quantomeno, imprevedibili a prescindere, poiché si trattava del risultato di una query, eseguita in PHP per giunta. Alla fine, giunsi a questo tipo di implementazione:

- Il carosello, di dimensione iniziali 5 e poi ridotte a 4, esisteva sin dall'inizio. Ovviamente al suo interno non aveva niente, e pertanto il div che lo contiene, di default, è invisibile;
- Quando veniva richiesta una query di carte, veniva lanciato un Ajax a una libreria php che avrebbe gestito la query. Nel mentre, un'immagine di caricamento sarebbe apparsa sul lato sinistro (l'immagine in realtà è sempre lì presente, ma è invisibile. L'invio di una query "svelava" l'immagine dando l'illusione di comparire solo a richiesta;
- Quando l'Ajax terminava, e il risultato della query veniva dato:
    - Veniva nuovamente oscurata l'immagine di caricamento;
    - Una "galleria" inviisibile veniva caricata con le varie immagini: per ogni carta ritornata veniva generato un div che conteneva immagine, nome e % di corrispondenza;
        - Cliccare su una di queste carte avrebbe richiamato la funzione di aggiunta della carta al mazzo (se già non vi era presente). Tenendo premuto SHIFT durante l'operazione, la carta sarebbe stata

aggiunta, di default, al Sideboard piuttosto che al mazzo principale;

- Inizialmente, le carte usavano un'immagine "standard", ossia il dorso di una carta.

- Viene invocata un'API offerta dai ragazzi dell'Applab, che avrebbe interrogato il database principale e recuperato, per ognuna di quelle carte, l'immagine delle stesse tramite varie chiamata Ajax. A quel punto, l'immagine ritornata (se c'era) aggiornava il dorso della carta, cambiandolo con l'immagine effettiva desiderata
- Il carosello viene pertanto aggiornato con le prime 4 carte (o con l'unica carta in caso di ricerca top1) della galleria. Se il numero di risultati è superiore a 3, vengono anche mostrati dei pulsanti di scorrimento che permettono di scorrere tale carosello;
- Il carosello veniva rivelato. Altre funzioni aggiuntive avrebbero svelato un secondo carosello, quello delle carte del mazzo, che segue una logica molto simile.

Una volta riuscito ad implementarlo ci fu da considerare l'idea di mettere un modo per riconoscere, a vista, quando il carosello stava rincominciando – doveva essere possibile infatti leggerlo in modo ciclico, e in ambo i versi, ma doveva anche essere possibile capire quando finiva in un modo più appariscente del notare che la % di corrispondenza stava incrementando invece che decrementando. La richiesta era una necessità, dovuta sia dall'assenza al tempo della maggior parte delle immagini, sia all'instabilità della prima API: le immagini venivano ritornate sotto forma di blob, e quando ne venivano chieste troppe, l'eccessiva mole di dati faceva scattare degli "allarmi" PHP che causavano la morte del demone e conseguente crash del server in cui l'API risiedeva.

Colorare dinamicamente una serie di righe o di colonne è abbastanza facile, un po' meno quando bisogna eseguire dei gradienti di tali colori e non si sa arbitrariamente quanto è il massimo – evidentemente 5 sfumature diverse potevano andar bene per la ricerca di una top1 o una top5, ma erano sin troppo insufficienti per una top10. Cosa ancora peggiore, non c'era un modo di aumentare la "lucentezza" di un colore, anche se avessi trovato come, dinamicamente: calcolare una % di luminosità, basata sulla % di corrispondenza (la carta con meno corrispondenza avrebbe avuto una sfumatura di colore più bianca di quella con più corrispondenza).

In genere si utilizza il formato RGB per calcolare un colore. Supponendo di prendere delle sfumature di grigio puro, ciò significa disporre di al più 16 sfumature "pure" (da #000000 a #FFFFFF, se vogliamo includere bianco e nero come sfumature di grigio, altrimenti 14), un valore insufficiente per quando si trattava di ricercare una top25. Non avevo desiderio di creare un mini-script che convertisse un numero da base decimale ad esadecimale, ritenendo che un valore % sarebbe stato più che sufficiente per fare quel che desideravo. Compresi che quel che mi serviva era una rappresentazione diversa del colore. Il passaggio da RGB a HSL era esattamente quel che cercavo: un singolo colore, con saturazione fissa ma luminosità diversa e, per giunta, decisa da un valore percentuale! In questo modo, il carosello ottenne le sfumature di colore deciso, non prima di aver forzato il valore di luminosità a una soglia minima del 30% - a 0% qualunque colore HSL è nero, e non volevo iniziare da quello- e quindi lavorato un po' per ri-calcolare le varie % al fine di cominciare da tale soglia e ripartirsi proporzionatamente.

3.3.3.3) Sideboard e Main Deck

Sebbene non sia obbligatorio, un giocatore ha la facoltà di crearsi un Sideboard, ed era pertanto giusto permettere la sua generazione durante la ricerca delle varie carte. Bisogna ricordarsi quanto segue:

- Una carta può essere presente nel Mazzo Principale, Sideboard o in ambo i mazzi;
- Un Sideboard era comunque necessario se un giocatore voleva realizzare una decklist Commander (dove il Sideboard è il Commander stesso)

Una prima risoluzione stava nel mettere un secondo contatore, che indicasse la quantità nel Sideboard. Non avevo i mezzi per poter far rispettare il "limite delle 4 carte" (il database non permetteva di distinguere lo status di una carta come "Terra Base", tipologia di carta per cui questa regola non viene rispettata), però potevo far rispettare il limite delle 15 carte del Sideboard: un eventuale inserimento di sedicesima carta sarebbe stato ignorato.

Tuttavia, si desiderava fare in modo che una carta potesse essere inserita direttamente nel Sideboard, senza passare per l'inserimento default nel mazzo principale. Come soluzione fu scelta quella di abilitare l'inserimento nel sideboard di default nel caso venga premuto un pulsante. In questo caso, il pulsante SHIFT.

Diventava pertanto necessario "sniffare", ovvero rimanere in ascolto della tastiera, per poter far scattare le funzioni ad hoc di inserimento nel Sideboard. JavaScript/JQuery offriva delle soluzioni native, ovvero:

```
$([oggetto bindato]).click(function(evt){if (evt.ShiftKey)
{[azione]}});
```

dove “evt” sta per “evento”, ossia un'azione generica rilevata a livello HTML, comunicata poi a JQuery che lo tratta con i propri metodi.
Ci stavano però due tranelli:

- L'evento doveva essere passato come parametro. In principio, il click di una carta faceva scattare direttamente la funzione di inserimento nel mazzo principale;
- il metodo “ShiftKey”, così come molte altri suoi simili, fa parte di uno standard che non tutti i browser supportano.

Fortunatamente, fu possibile trovare una soluzione ad ambo i problemi in poco tempo. Rispettivamente:

- Generazione di una “funzione router”. La pressione/depressione del tasto Shift settava a true/false una variabile JavaScript globale. Quando si cliccava una carta, la funzione chiamata era questa “router” che reindirizzava poi alla funzione di inserimento nel mazzo principale o nel Sideboard se in quel momento tale variabile era settata rispettivamente a True o a False (quindi durante la pressione del pulsante);
- Utilizzare l'attributo event.which, il cui valore era un numero corrispondente al codice di tastiera del pulsante premuto. Nel caso di Shift, il codice è 16

La soluzione di event.which era sicuramente portabile, ma non completamente: sebbene raro, è possibile che i codici di tastiera cambino tra i vari browser. Tutti i browser più importanti, da

Chrome a Safari, tuttavia, riconoscono questo codice correttamente.

3.3.3.4) Ricerca semantica

Questa è una funzione aggiuntiva richiesta durante la creazione del configuratore, *ossia permettere l'inserimento di una carta arbitraria all'interno del mazzo, invece che partire da una delle 25 carte più usate*. Inoltre, doveva essere inserita una funzione di auto-completamento, così da sveltire la ricerca della carta desiderata.

In genere, il processo standard prevede quanto segue:

- Una Textarea viene dedicata all'inserimento della carta. Ad essa viene legata una funzione JavaScript che si eccita ad ogni keyUp;
- la funzione controlla che:
  - Il tasto appena rilasciato sia un carattere e/o un numero;
  - Siano stati inseriti un certo numero di caratteri, in genere 3;
- Se i controlli vanno a buon fine:
  - Chiamata Ajax a una libreria PHP, che sbriga una query sul server per ottenere i vari matches;
  - Una volta ritornati tali matches, viene popolato un div con i risultati suggeriti, oppure viene auto-completata la textarea con il primo risultato in ordine alfabetico (con l'accortezza di "selezionare" la parte aggiunta, in modo da permettere un'agevole sovrascrittura in caso di suggerimento "errato")

Per quanto rispettabile come scelta, la scartai dopo aver tenuto in considerazione il preoccupante overhead che avrebbe comportato il

lancio di una query a ogni pulsante premuto. Piuttosto, volevo cercare una soluzione full-JavaScript. E riuscii a trovarla:

- In principio, quando la pagina veniva caricata (o ri-caricata), viene eseguita una query via PHP per ottenere il nome di tutte le carte, e il risultato viene poi mandato in una variabile di sessione JavaScript, aggiornandone il valore;
- Quando si inizia a scrivere, indipendentemente da quanto si è scritto:
  - Viene aggiornato un "timer" di circa 200 millisecondi, riportandolo al valore iniziale. Questo permette sia a chi scrive veloce di non far scattare suggerimenti "inutili", ma anche allo stesso browser di non "auto-sovrascriversi" gli eventi keyUp comportati da una digitazione rapida;
  - Se il timer va a 0, cioè ci si è fermati per quella frazione di tempo:
    - Viene fatta partire una funzione JavaScript che prende il contenuto della Textarea;
    - A quel punto, inizia a ricercare nell'array del nome delle carte il primo match in ordine alfabetico, ossia quel nome di carta con i primi N caratteri esattamente identici a quelli della stringa nella Textarea, dove N è la lunghezza di suddetta stringa
      - Se trova tale match, lo inserisce nella Textarea, autocompletandola e avendo cura di selezionare quanto è stato così aggiunto;
    - In aggiunta, inizia a ricercare per tutto l'array di nomi i nomi che hanno quanto digitato come sottostringa generale (quindi non necessariamente che inizino per quanto scritto);

- Questo è permesso cercando quei valori in cui l'indexOf della sottostringa nella Textarea è diverso da -1, codice da interpretare come "sottostringa non trovata";
- Se ne trova, un limite fissato – in questo caso di 10 – andrà a popolare una tendina di suggerimenti aggiuntivi;

Per fare un esempio, se si ricerca la parola "Urza", come risultato l'autocompletamento scriverà "Urza's Factory" (la prima in ordine alfabetico), ma tra i suggerimenti comparirà anche "Glasses of Urza" , poiché contiene "Urza" nel suo nome.

Il pulsante di inserimento, in principio, invocava il metodo di inserimento della carta. La versione attuale, tuttavia, sovrascrive il carosello con uno da una carta, che mostra la singola carta che si sta cercando. L'inserimento avviene, pertanto, come di norma cliccandoci sopra.

Questo sistema di autocompletamento è altresì utilizzato per la ricerca del nome dei mazzi. Quando si va a mettere il nome di un mazzo, vengono suggeriti i primi 5 mazzi omonimi e completato con il primo mazzo in ordine alfabetico corrispondente. Questo permette:

- In caso di caricamento: di facilitare la ricerca di certi mazzi;
- In caso di salvataggio: di sapere immediatamente che un nome è già stato preso, poiché lo stesso giocatore non può creare due mazzi omonimi.

3.4) La funzione di Reset (ex reset.PHP)

Un tempo, nella pagina principale, era presente un piccolo link, quasi invisibile, il cui accesso dovrebbe essere permesso solo a degli amministratori. Esso portava a una pagina sgraziata tanto quanto il suo scopo: la cancellazione completa del database.

Nelle prime fasi di testing del webscraper, infatti, mi ero trovato più volte nella necessità di dover resettare l'intero database per via di inserimenti parziali e sbagliati. Trovai, però, inefficace utilizzare un semplice drop di ogni tabella, in quanto non resettava l'auto-increment e andava in conflitto con alcuni vincoli di restrizione. Alla fine, sviluppai una sequenza di query mySQL, nulla più che dei TRUNCATE con annessa opzione di ignora-vincoli uno dietro l'altro, che avrebbero cancellato volendo anche più in fretta le tabelle e, al contempo, resettato l'auto-increment. Per molto tempo tale funzione è rimasta nella pagina webscraper.php, poi fu concordato che era già tanto se veniva permesso a un utente di cancellare un singolo evento in modo deliberato, e questa funzione fu relegata in questa pagina, reset.php. Successivamente, il committente cambiò idea, e la funzione ritornò di nuovo nel webscraper, dopo essersi assicurati che solo gli admin potevano averci accesso.

Al tempo, la pagina era provvista di un testo a monito di quel che si stava facendo, una checkbox che fa da surrogato di captcha stile "non sono un robot" e un "pulsante rosso" che resetta tutto. Riportata su webscraper, fu mantenuto solo un pulsante e la checkbox simil-captcha. Poichè il caricamento dei primi 13.000 eventi ha richiesto circa 3 giornate di costanti caricamenti in gruppi da 50, mi auguro non ci sia mai davvero necessità di

ricorrere a tale funzione – e soprattutto che vengano eseguiti dei backup periodici del database in caso di attacchi malintenzionati.

4) Sviluppi futuri

Nonostante gli sforzi fatti, ci sono ancora molte problematiche da risolvere in questo lavoro, alcune dovute da mancanza di tempo, altre perché richiedono competenze avanzate per la mia formazione, altre temo proprio per assenza di mezzi di implementazione. Questa che segue è una lista dei vari "to do".

4.1) Un codice statico per un sito dinamico

Questo problema chiaramente si propone solo su webscraper.php, e indica un problema comune di qualsiasi XPath. Generai questo sito nel periodo di transizione di Python dal 2.x al 3.x, e nel periodo di transizione di mysql a mysqli, pertanto parve appropriato al fato che la cosa coincidesse anche con un revamp totale grafico (e di codice) dell'intero sito MTGTOP8. Quando vidi tale cambiamento, non potei che mettermi le mani tra i capelli, poiché sapevo già cosa significava: gli XPath che avevo calcolato al tempo non erano più validi – le posizioni che cercavo non c'erano più!

Per mia fortuna, il cambiamento al codice fu minimo, ma comunque dovetti perdere un giorno per ri-visionare buona parte del sito e ri-calcolare i vari XPath aggiustati. Era però chiara una problematica di questo approccio ad XPath: un qualsiasi cambiamento, anche minimo, del codice HTML avrebbe potuto seriamente minare l'efficacia di un XPath. E se anche uno di questi valori non è più al suo posto lo script rischia di perdere informazioni o, peggio ancora, di bloccarsi ed eseguire inserimenti parziali, o nulli addirittura. Rendendo tutto quel che è stato fatto fino adesso totalmente inutile.

Pertanto, nello sciagurato caso MTGTOP8.com dovesse nuovamente cambiare grafica, sarà necessario ritornare a ricalcolare XPath. Un problema, purtroppo, che temo non possa rendere automatizzabile.

4.2) Un modello noSQL?

MySQL, o SQL *(Structured Query Language)* è indubbiamente uno dei più famosi e utilizzati linguaggi di relazione database impiegati. Tuttavia, sono possibili altri approcci:

- Uso di un linguaggio per database relazionali diverso (es. PostgreSQL);
- uso di un database "noSQL", cioè non-relazionale.

L'interesse cade su quest'ultimi, che potrebbero essere più appropriati data la peculiarità offerta da questo database.

Figura 35: Il logo di Oracle NoSQL Database

Oracle NoSQL Database: [37] è un Database Chiave-Valore, meglio noto anche come "Dizionario" o "Hash". Se, nella versione RDB, DataBase Relazionale, un dato viene letto come una tabella con campi e tipi di dati ben definiti, nel Dizionario esso è trattato come una singola collezione con campi diversi per ogni record. Nel caso di Oracle NoSQL Database, questo si traduce come una lunga sequenza di record, o righe, tutte con chiave unica e un valore di lunghezza arbitraria interpretato poi dall'applicazione. Lo stesso valore ha più chiavi, che a loro volta possono avere sub-chiavi, con

[37] https://en.wikipedia.org/wiki/Oracle_NoSQL_Database

un'organizzazione simile a quella di una path di un file system. Lo Schema è definibile via JSON.

- PRO
  - Enorme diminuzione dello spazio occupato
  - Di facile comprensione per un programmatore abituato alla programmazione ad oggetti
  - Dispone di una seconda fila di indici
- CONTRO
  - Prestazioni inferiori alla media
  - Mancanza di standard di riferimento

Figura 36: Il logo di Apache Cassandra

Apache Cassandra: [38] è un Database ibrido tra Chiave-Valore e Orientato a Colonne. Invece dello standard SQL, utilizza lo CQL (Cassandra Query Language), che utilizza le COLUMNFAMILY al posto delle TABLES (usate, piuttosto, per l'organizzazione interna delle righe) e le PARTITION KEY per organizzare i vari Cluster di dati. Questo permette a un record di avere colonne diverse rispetto ad altre della stessa Columnfamily. Ogni tabella appartiene ad un cluster diverso, e la loro distribuzione tra di essi può essere eseguita in modo da tenere vicine quelle simili (OPP, Order Preserving Partitioner) oppure messe alla rinfusa (RP, Random

[38] https://en.wikipedia.org/wiki/Apache_Cassandra

Partitioner), a seconda di quanto si vuole equilibrato un caricamento.

- PRO
  - Prestazioni di qualità eccelsa;
  - Esiste un porting in C++, di nome ScyllaDB;
  - Flessibilità nella rappresentazione dei dati;
- CONTRO
  - Prestazioni di lettura e scrittura direttamente proporzionali al numero di clusters;
  - Inconsistente a livello riga;
  - Impossibile eseguire Join e Subquery (ma usa altri costrutti per compensare tale mancanza);

Figura 37: Il logo di OrientDB

OrientDB: [39] è un Database full ibrido, poiché permette l'uso anche di modelli come quello a Oggetti o a Documento. Tutte le relazioni sono trattate come se fosse un Database a Grafico, con connessioni dirette ai record. Proprio per la sua totale flessibilità, permette l'uso di query in SQL oltre che Gremlin per comunicare con esso. Ogni record ha anche il suo ID personale (da non intendere come la chiave primaria necessariamente), che viene poi organizzato in un B-Tree (un albero con capacità di auto-equilibrarsi, simili a un binario, ma che permette ai nodi di avere più di due figli), rendendo le procedure di lettura, rimozione e

[39] https://en.wikipedia.org/wiki/OrientDB

scorrimento dell'albero risolvibili in O(logN) (lo scorrimento addirittura in O(1))

- PRO
  - Totale e completa flessibilità sulla rappresentazione dei dati;
  - Transazione da SQL a noSQL facile e facilitata;
  - Comunità online attiva e in continuo sviluppo;
- CONTRO
  - Alcune funzioni sono a pagamento;
  - Pensato per lo più per grandi strutture dati;
  - Richieste di memoria eccessive;

5) Conclusioni

5.1) Conclusioni

Terminata l'ultima implementazione, mi fermai a guardare quanto avevo fatto:

- Avevo creato una utility di web scraping:
  - Appoggiandomi su PHP/JavaScript, invocavo tramite exec alcuni script Python che gestivano interamente il processo di web scraping, con informazioni immagazzinate in DB MySQL;
    - Tali script leggono le pagine tramite libreria lxml, e urllib dopo aver scartato altre scelte come BeautifulSoup, e leggendo l'albero del DOM ottengono le informazioni tramite calcolo dell'XPath;
    - Tali script si connettono a MySQL tramite libreria MYSQL-CONNECTOR-PYTHON, dopo aver scartato scelte come pymssql e PyMySQL, richiedendo l'uso di una VM Python ad hoc che la implementasse per poter permetterne l'uso anche in versioni di Debian obsolete
    - Alcuni aggiustamenti hanno permesso la comunicazione di dati in formato pseudo-UTF-8 nonostante setting diverso di tale macchina virtuale, che costringeva all'uso di ASCII;
    - Gli scripts mantenevano comunque un certo livello di controllo appoggiandosi a MySQL;
  - Tramite uso di PThread, e relativo aggiustamento alla configurazione di PHP, era possibile lanciare più scripts nello stesso momento, limite dato solo dalle capacità del server;

- Richieste iterate periodicamente controllavano l'avanzamento dei processi, aggiornando a video tramite JavaScript delle tabelle. Sono anche presenti funzioni aggiuntive quali un sistema di tooltip o altre utility di gestione database, come la cancellazione di eventi o inserimenti parziali eseguiti per debug;

- Avevo creato un configuratore di gestione di mazzi di Magic: The Gathering:
  - La creazione del mazzo, di base, usa un approccio statistico, basandosi sul concetto di "fama" o comunque di "top x più usate in combinazione":
    - Tramite una query MySQL dal costo di O(2N), ove N è il tempo di lettura di una tabella di dimensioni notevoli (almeno 2 milioni di records), vengono restituite le prime X carte che sono state utilizzate assieme a una tupla di carte (inclusi i singleton), oppure la topX delle carte più usate in generale se la tupla è vuota. Tale query fa uso di una tabella temporanea per inserire i risultati delle varie subquery, per sfruttare quanto più possibile l'ottimizzatore MySQL;
      - Le prestazioni della query dipendono dal grado di "fama" di ogni singola carta (più una carta è famosa, meno mazzi saranno filtrati e maggiore il carico di lavoro conseguente) e, in definitiva, dalle prestazioni del server;
      - L'uso dell'ottimizzatore MySQL permette di accorciare i tempi delle richieste più famose o comunque già eseguite recentemente;
  - Presenta comunque una possibilità di ricerca di carte in modo semantico, tramite apposita funzione:

- La funzione suggerisce in tempo plesiocronico 10 nomi di carte che contengono la stringa inviata, usando come suggerimento iniziale (se esiste) un nome che inizia con la stringa ricercata. La ricerca, in genere svolta via query MySQL gestite via Ajax e libreria PHP apposita, viene invece eseguita utilizzando un array JavaScript locale, di fatto diminuendo l'overhead in cambio di dati non aggiornati all'ultimissimo minuto;

- La gestione del mazzo consente di poter aggiungere carte nel mazzo principale e nel Sideboard a piacimento, senza rispettare le regole del gioco (tranne quella del massimo 15 carte nel Sideboard) per semplici mancanze di strutture dati appropriate:
  - Una volta che il Main Deck raggiunge dimensione 60, il pulsante di salvataggio diventa attivo. Se l'utente non ha già registrato prima di allora un mazzo omonimo, la decklist viene salvata;
  - È altresì possibile caricare decklist già salvate. Eventuali cambi di tale decklist saranno salvate come nuovo mazzo (quindi con nuovo nome). Per la ricerca del nome del mazzo si utilizza lo stesso sistema di ricerca semantica impiegata nelle carte, limitando stavolta il numero massimo di risultati suggeriti a 5;
- Le carte sono mostrate in carosello di dimensione 4, sia per le carte suggerite, sia per la decklist visiva del mazzo:
  - Cliccando sulle carte suggerite, in combinazione con la pressione di determinati tasti, è possibile aggiungere una carta al mazzo o al sideboard se già non è lì presente;

        - Le carte mostrano altresì il loro grado di “fama” e sono ordinate per esso. Il limite massimo di risultati così ricercabili è in genere 10 ma può essere incrementato fino a 25;
        - Il caricamento delle immagini delle carte è gestito da un API impiegata nel grande progetto di cui questa tesi fa parte. La disponibilità di tali immagini dipende sia dalla disponibilità del server in cui il DB di immagini risiede, sia dalle immagini effettivamente caricatevi dentro;
    - Presenta anche alcune statistiche generali, mostrate in scorrimento laterale a fondo pagina;
- Avevo generato alcune pagine ausiliarie al progetto:
    - Una pagina index.php, utilizzata per l’accesso alle due funzioni del configuratore;
    - Una pagina login.php, dove gli utenti potevano accedere, e dove l’accesso veniva ulteriormente filtrato in caso di accesso all’utility di web scraping;

“Da contratto”, il mio lavoro era da considerarsi terminato.

Ma sentii ben poco il senso di appagamento o di contento. Piuttosto, sapevo di aver concluso, dopo tutto quel tempo, nulla più che una base, una versione 1.0 di qualcosa con potenziale di ottimizzazione ed ampliamento. Ma d'altronde, non è forse questo il destino di tutti gli applicativi, o i configuratori in questo caso? Le richieste dell'utente finale di un programma sono sempre mutevoli, viene domandata l'implementazione di nuove funzioni mentre altre diventano deprecate, standard che rivoluzionano ambienti di programmazione e documenti internet nascono e muoiono – sotto questo punto di vista, un programma è molto simile ad una pianta:

la sua crescita e sviluppo dipendono dall'abilità del giardiniere, dalla bontà del terreno e dalla buona fortuna di non essere mai testimoni di tremendi cataclismi.

La sezione degli sviluppi futuri ha dato una panoramica dei grandi lavori ancora nella “to-do list”, ma ce ne sono molti altri, più piccoli, che possono perfezionare ancora di più quanto fatto. Non ultimo, la possibilità di collegare effettivamente questo configuratore al grande database principale – cosa che permetterebbe una creazione di mazzi sempre più adatta alle esigenze di giocatori professionisti, o anche solo di permettere una funzione di registrazione una volta che il progetto sarà esposto al pubblico generale.

Non esiste davvero un programma o un configuratore “impeccabile”. Molto è stato fatto, e molto rimane ancora da fare. Al programmatore non resta che lasciare un biglietto da visita, al suo ex-committente, in segno di disponibilità al supporto tecnico… e speranza di poter tornare a mettere a disposizione le proprie abilità per il bene degli altri.