

Sito web per la gestione di un casinò

1	Introduzione.....	3
1.1	Informazioni sul progetto.....	3
1.2	Abstract	3
1.3	Scopo	4
Analisi.....		4
1.4	Analisi del dominio	4
1.5	Analisi e specifica dei requisiti	4
1.6	Pianificazione	7
1.6.1	Stesura del Gantt	8
1.6.2	Analisi.....	8
1.6.3	Progettazione	9
1.6.4	Implementazione.....	10
1.6.5	Protocolli di test.....	12
1.6.6	Documentazione di progetto	13
1.7	Analisi dei mezzi.....	14
1.7.1	Software	14
1.7.2	Hardware.....	14
2	Progettazione	15
2.1	Design dell'architettura del sistema	15
2.2	Design dei dati e database.....	15
2.3	Design procedurale	17
2.3.1	PHP.....	17
2.3.2	Java.....	20
3	Implementazione	23
3.1	Front-end (pagine web – HTML + CSS)	23
3.2	Back-end (PHP)	35
3.2.1	Classe user:	35
3.2.2	Classe Database:.....	37
3.2.3	Classe SendMail:	40
3.2.4	Esempio utilizzo classi:	41
3.3	Test front-end (Selenium + JUnit)	41
3.4	Test automatizzati con Jenkins	44
3.5	Test back-end (PHPUnit)	44
3.6	Database (MySQL).....	44
3.6.0	Creazione database	44
3.6.1	Room.....	44
3.6.2	Game	45
3.6.3	Media type.....	45
3.6.4	Media	45
3.6.5	Game media.....	46
3.6.6	Gender	46
3.6.7	Promotion.....	46
3.6.8	Promotion media	46
3.6.9	User type.....	47
3.6.10	User.....	47
3.6.11	Promotion user.....	47
3.6.12	Room media.....	48
4	Test.....	48
4.1	Protocollo di test.....	48
4.2	Risultati test	52
4.3	Mancanze/limitazioni conosciute.....	53
5	Consuntivo.....	53
6	Conclusioni	53
6.1	Sviluppi futuri.....	53
6.2	Considerazioni personali.....	53
7	Bibliografia	54
7.1	Sitografia	54
	Allegati	54

1 Introduzione

1.1 Informazioni sul progetto

Autore: Matan Davidi, Thor DUBLIN, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Scuola: Arti e Mestieri Trevano

Classe: I3AA

Anno scolastico: 2019

Sezione: Informatica

Materia: Modulo 306

Docenti responsabili: Massimo Sartori

Data di inizio: 13.02.2019

Data di consegna: 22.05.2019

1.2 Abstract

This document contains the documentation of the realization of a casino's management software. This software has to manage the casino's users, its games, its rooms and the promotions it can offer to its users. Also, games and rooms can have images or videos to accompany them.

The software's realization has to implement a system of continuous integration and automatic testing that works alongside a version control tool, such as GitHub, so that it doesn't allow the team that is working on it to push something to the repository without first checking that it passes every test for quality control. Basically, this way, nothing that breaks the functioning of the application can be pushed to production.

Also, a user's name, surname, address, house number, zip code, city, email address, phone number and gender registered in the system's database. Each game has a name and a room where it is played, each room a location where it is and, for each media file, the URL, game or room it represents, and its type (picture, video, ...) are stored. Finally, the promotions are handled by storing the message they show and the user they are shown to.

The quality controls consist in three subcategories:

- Emails (checking if the system is able to send an email)
- Database (testing if the system can connect to the database, testing if the system cannot connect with invalid credentials, testing if the system is able to insert a new user, testing if the system cannot insert a new user with an invalid format, testing if the results returned from a query are correct, testing if the system does not return information requested by a malformed query)
- Users (controlling that a new user can be created before inserting it into the database, controlling that a new user cannot be created if its data is invalid, controlling if each value inserted for a new user is valid and not invalid)

1.3 Scopo

Lo scopo di questo progetto è quello di creare un'applicazione web che semplifichi la gestione di un casinò in ogni suo aspetto: gli utenti, i giochi, le sale e le promozioni.

Inoltre un altro obiettivo è quello di insegnarci a utilizzare sistemi di test e di integrazione continua che vengono usati anche in grandi aziende come Selenium e Jenkins in modo da poterli utilizzare in futuro quando ci ritroveremo a lavorare in una ditta vera.

Analisi

1.4 Analisi del dominio

Il dominio per questa applicazione è pressoché inesistente, in quanto l'applicazione per la gestione del casinò deve venir fatta da capo partendo dai requisiti del cliente e non esistono applicazioni che potremmo usare come modello dalle quali ispirarci.

1.5 Analisi e specifica dei requisiti

ID: REQ-001	
Nome	Sito web per la gestione di un casinò
Priorità	1
Versione	1.0
Note	
<i>Sotto-requisiti</i>	
001	Si deve poter navigare senza imprevisti per tutta la pagina web.

ID: REQ-002	
Nome	Gestione degli utenti
Priorità	2
Versione	1.0
Note	
<i>Sotto-requisiti</i>	
001	Per ogni utente devono essere memorizzati il nome, cognome, l'indirizzo, il numero civico, la città, il NAP, l'indirizzo email, il numero di telefono, il sesso e una password.

ID: REQ-003	
Nome	Gestione giochi
Priorità	2
Versione	1.0
Note	
<i>Sotto-requisiti</i>	
001	Per ogni gioco bisogna gestire il nome e la sala nella quale si trova.

ID: REQ-004	
Nome	Gestione sale
Priorità	2
Versione	1.0
Note	
<i>Sotto-requisiti</i>	
001	Per ogni sala deve essere gestita la posizione all'interno casinò.

ID: REQ-005	
Nome	Registrazione e accesso
Priorità	3
Versione	1.0
Note	
<i>Sotto-requisiti</i>	
001	Deve essere possibile registrarsi come utente alla piattaforma.
002	I dati da che l'utente deve inserire sono i seguenti: Nome, cognome, via, numero civico, CAP, città, indirizzo email, numero di telefono, sesso, data di nascita, password
003	Deve essere eseguita la conferma che l'indirizzo email inserito sia valido tramite l'invio di un'email a quell'indirizzo con un link per attivare il proprio account.
004	Ci deve essere la possibilità di avere utenti amministratori che possano aggiungere, modificare e cancellare dei dati dal database.
005	Gli utenti possono essere di diversi tipi, in modo che le promozioni possano venire mostrate a una sottocategoria di utenti.

ID: REQ-006	
Nome	Utenti amministratori
Priorità	3
Versione	1.0
Note	
<i>Sotto-requisiti</i>	
001	Gli utenti amministratori devono poter aggiungere, modificare e rimuovere dati relativi agli utenti.
002	Gli utenti amministratori devono poter aggiungere, modificare e rimuovere dati relativi alle sale.
003	Gli utenti amministratori devono poter aggiungere, modificare e rimuovere dati relativi ai giochi.
004	Gli utenti amministratori devono poter aggiungere, modificare e rimuovere dati relativi alle promozioni.
005	Gli utenti amministratori devono poter aggiungere immagini da mostrare relativamente a un gioco, una sala o una promozione.

ID: REQ-007	
Nome	Gestione promozioni
Priorità	3
Versione	1.0
Note	
<i>Sotto-requisiti</i>	
001	Devono essere gestite delle promozioni da mostrare a uno o più utenti sotto forma di pagine web.
002	Ogni promozione deve contenere un titolo, una descrizione e una o più immagini o video.
003	Deve essere gestito quale promozione viene visualizzata da quali tipi di utenti.

ID: REQ-008	
Nome	Protocolli di test e integrazione continua
Priorità	1
Versione	1.0
Note	
<i>Sotto-requisiti</i>	
001	Devono essere implementati dei protocolli di test che permettano di verificare il funzionamento totale dell'applicazione.
002	I protocolli di test devono essere avviati ogni volta che viene eseguito un push verso il repository di GitHub.

003	I protocolli di test comprendono la verifica che le email vengano inviate correttamente
004	I protocolli di test comprendono la verifica della corretta connessione al database
005	I protocolli di test comprendono la verifica della validità dei dati inseriti dall'utente in fase di registrazione

1.6 Pianificazione

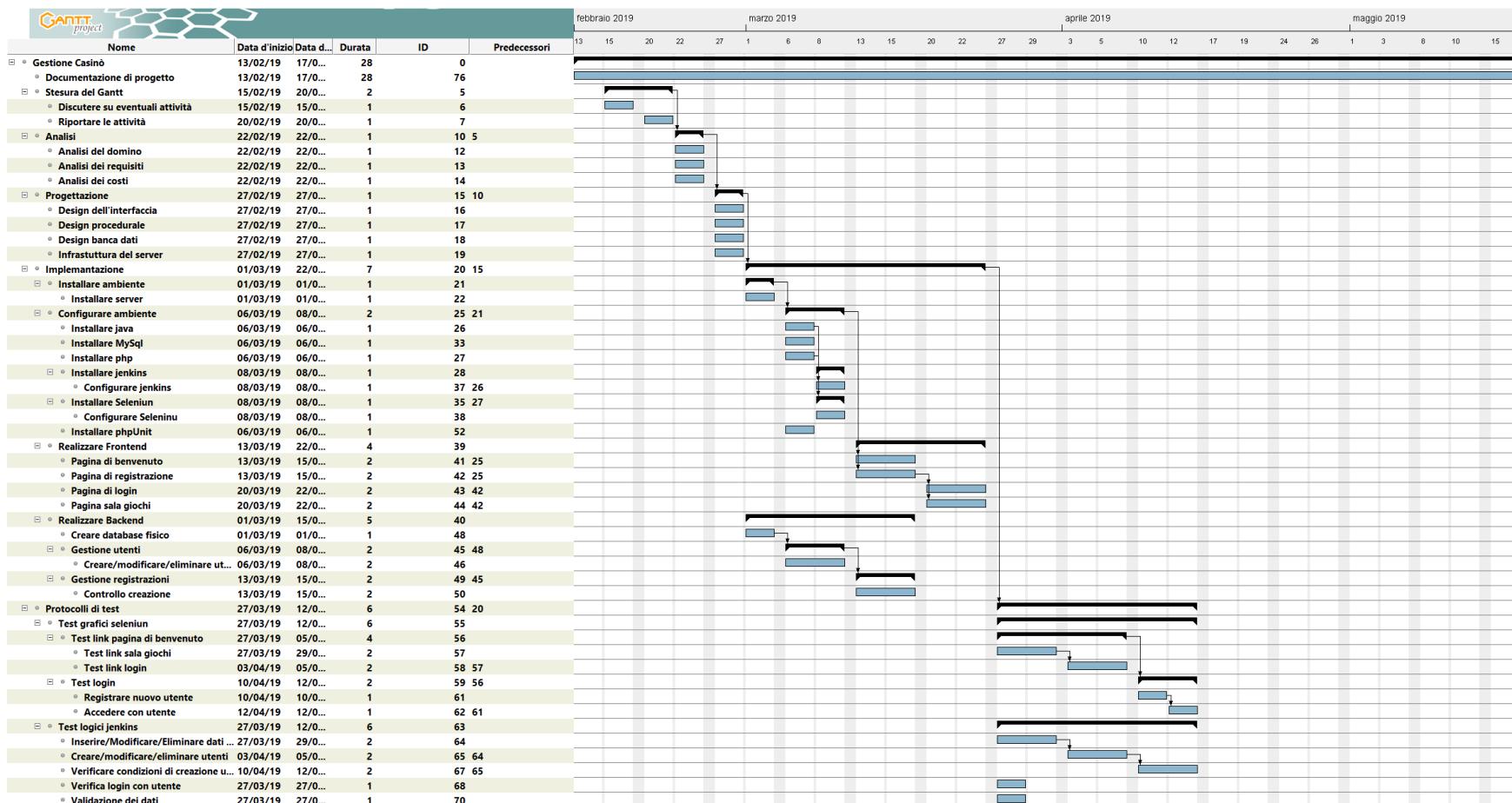


Figura 1: diagramma di Gantt utilizzato per pianificare il progetto

La pianificazione si divide in 5 fasi distinte: Stesura del Gantt, Analisi, Progettazione, Implementazione, Protocolli di test; ognuna delle quali si suddivide nuovamente in attività.

Gestione casinò

1.6.1 Stesura del Gantt

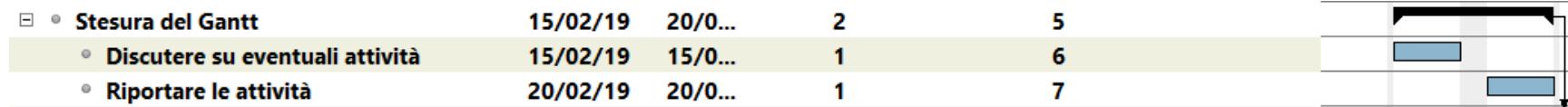


Figura 2: Attività "Stesura del Gantt"

La stesura del Gantt è quell'attività che porta ad avere una pianificazione del progetto e consiste in due operazioni:

- Discutere su eventuali attività, che consiste nel discutere tra i componenti del gruppo i passi necessari per riuscire a realizzare il progetto.
- Riportare le attività, che implica creare la pianificazione del progetto grazie a uno strumento come GanttProject.

1.6.2 Analisi



Figura 3: Attività "Analisi"

L'analisi del progetto in questo caso è consistita nell'informarsi ognuno sui nuovi software che si sarebbe ritrovato a utilizzare. Questo significa che Matteo Forni si è informato sull'installazione e l'utilizzo di Jenkins, Carlo sull'utilizzo di PHPUnit, Thor sull'utilizzo di Selenium e JUnit. Inoltre abbiamo effettuato le seguenti operazioni:

- Analisi del dominio, ovvero l'analisi della situazione attuale prima della realizzazione del progetto che permette di valutare se ha senso realizzare il progetto.
- Analisi dei requisiti, che consiste nell'analizzare le richieste del cliente e stilare una lista di requisiti che l'applicazione deve soddisfare prima di poter essere consegnata al committente.
- Analisi dei costi, che implica un'analisi di costi e benefici del progetto in modo da definire se vale la pena dal nostro punto di vista creare l'applicazione o se i costi sono maggiori dei benefici.

Gestione casinò

1.6.3 Progettazione

<input checked="" type="checkbox"/>	<input type="radio"/>	Progettazione	27/02/19	27/0...	1	15	10				
	<input type="radio"/>	Design dell'interfaccia	27/02/19	27/0...	1	16					
	<input type="radio"/>	Design procedurale	27/02/19	27/0...	1	17					
	<input type="radio"/>	Design banca dati	27/02/19	27/0...	1	18					
	<input type="radio"/>	Infrastuttura del server	27/02/19	27/0...	1	19					

Figura 4: Attività "Progettazione"

La progettazione è consistita nell'organizzare l'implementazione del progetto in modo da dividere il lavoro all'interno del team e non avere problemi in cui due componenti stanno lavorando su cose contrastanti. Essa si divide in:

- Design banca dati, ovvero la progettazione del database tramite la realizzazione di un diagramma Entità/Relazioni e del relativo schema logico.
- Infrastruttura del server, ossia la progettazione di ogni componente che verrà poi installato sul server di produzione.

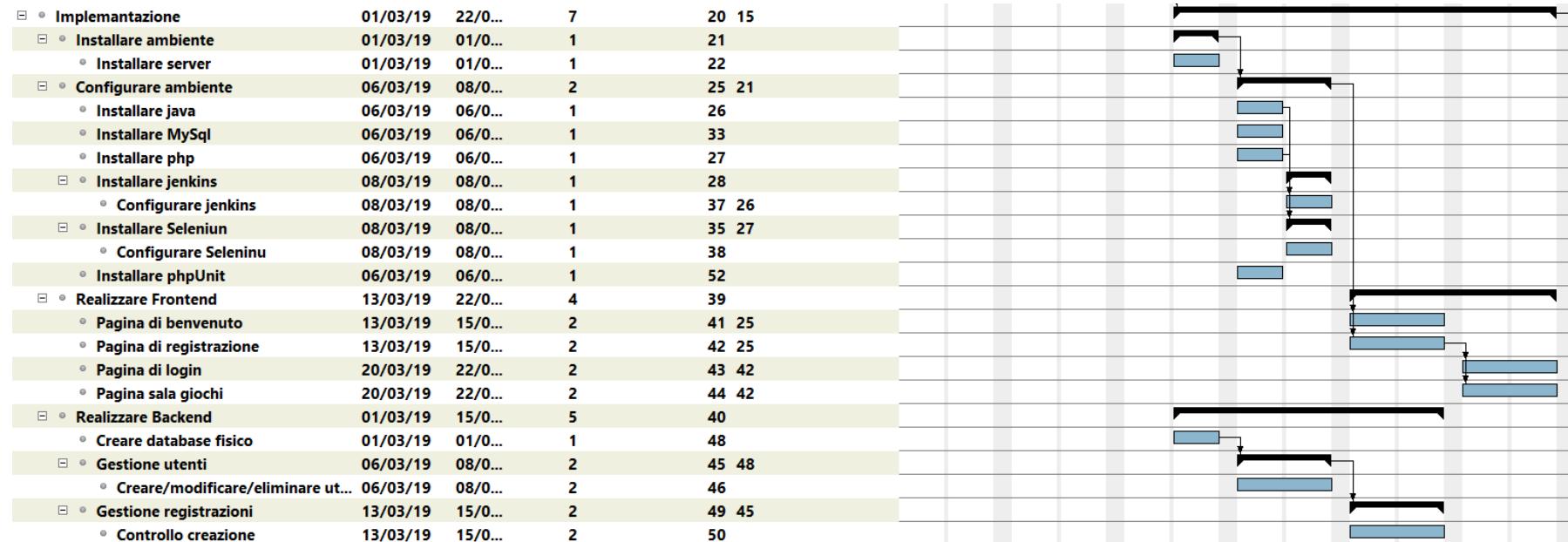
Gestione casinò
1.6.4 Implementazione


Figura 5: Attività "Implementazione"

L'implementazione è stata la parte più lunga del progetto oltre alla documentazione. È l'attività durante la quale abbiamo dovuto realizzare il progetto in ogni suo aspetto, basandoci sulla progettazione fatta nel punto precedente. Essa è divisa nelle seguenti operazioni:

- Installare ambiente, che a sua volta è categorizzato da:
 - o Installare server, ovvero l'installazione e la configurazione di tutti i componenti di base necessari per l'utilizzo del sistema operativo.
- Configurare ambiente, che si divide in:
 - o Installare Java
 - o Installare MySQL
 - o Installare PHP
 - o Installare Jenkins
 - Configurare Jenkins, per farlo funzionare in modo che ad ogni push sul repository di GitHub vengano eseguiti i protocolli di test che verifichino se il codice funziona anche dopo le modifiche descritte nel push.
 - o Installare Selenium
 - Configurare Selenium, per fargli eseguire i test dell'interfaccia grafica su "ordine" di Jenkins dopo ogni push.
 - o Installare PHPUnit
- Realizzare Frontend, ovvero realizzare le seguenti pagine web:

Gestione casinò

- Pagina di benvenuto
- Pagina di registrazione
- Pagina di login
- Pagina sala giochi
- Realizzare Backend, ossia la creazione del codice che permette di eseguire tramite interfaccia web delle operazioni. Più nello specifico:
 - Creare database fisico, la realizzazione del database MySQL sul quale si basa l'intera applicazione.
 - Gestione utenti, più nello specifico:
 - Creare/modificare/eliminare utente, il codice che permette di aggiungere, modificare e/o togliere un utente.
 - Gestione registrazioni
 - Controllo creazione, realizzazione della logica di controllo della creazione di una riga contenente i dati del nuovo utente all'interno del database.
- Protocolli di test (più dettagli nel capitolo Protocolli di test):
 - Test grafici Selenium
 - Test link pagina di benvenuto
 - Test link sala giochi
 - Test link login
 - Test login
 - Registrare un nuovo utente
 - Accedere con utente
 - Test logici Jenkins
 - Inserire/Modificare/Eliminare dati database
 - Verifica login con utente
 - Validazione dei dati
 - Creare/modificare/eliminare utenti
 - Verificare condizioni di creazione utenti

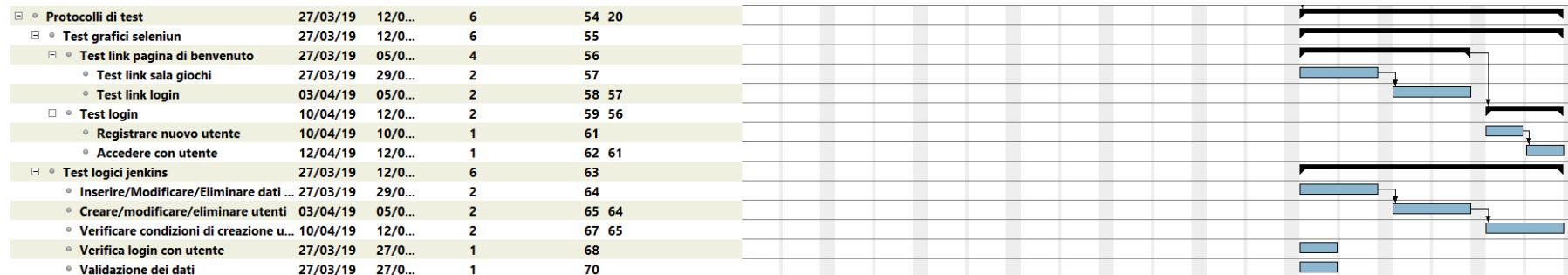
Gestione casinò
1.6.5 Protocolli di test


Figura 6: Attività "Protocolli di test"

I protocolli di test sono quelli che ci hanno permesso di verificare che il prodotto finale funzioni e che i requisiti imposti dal committente siano stati soddisfatti. Questi test sono divisi in:

- Test grafici eseguiti grazie a [Selenium](#), che a loro volta si scompongono in:
 - Test link pagina di benvenuto, ovvero accertarsi che i collegamenti alle altre pagine presenti nella schermata principale che si apre appena si accede al sito portino alle pagine a cui devono portare e non ci siano quindi problemi con la navigazione. Anch'essa si separa in:
 - Test link sala giochi, che verifica che i link nella pagina della sala giochi portino ai file giusti.
 - Test link login, che controlla che i link all'interno della schermata di login e registrazione puntino ai file giusti.
 - Test login, sviluppati in PHP e si dividono in:
 - Registrare nuovo utente, che si accerta che non ci siano problemi nella registrazione di un nuovo utente.
 - Accedere con utente, che appura che non ci siano errori nell'accesso alla piattaforma con un utente.
 - Test logici jenkins, ovvero i test che verificano la logica di funzionamento del programma in generale. Si suddividono in:
 - Inserire/Modificare/Eliminare dati database, la verifica di funzionamento di inserimento, modifica ed eliminazione di dati nel database in generale.
 - Creare/modificare/eliminare utenti, il controllo di creazione, modifica ed eliminazione di utenti dal database eseguendo i comandi MySQL.
 - Verificare condizioni di creazione utenti, l'accertamento che sia possibile creare utenti con delle certe condizioni per alcuni dati, per esempio l'email deve avere un certo formato, la password una certa lunghezza, ...
 - Verifica login con utente, verifica che il codice che dovrebbe permettere a un utente di accedere funzioni correttamente e prenda i dati dal database nel modo giusto.
 - Validazione dei dati, controllo che i dati non corretti vengano respinti dall'applicazione e che quelli validi vengano lasciati passare.

**1.6.6 Documentazione di progetto**

Infine c'è un'attività rimasta costante durante tutto il progetto: la documentazione. Infatti chi aveva dei tempi morti da riempire lo faceva documentando il progetto e aggiungendo informazioni al file che state leggendo in questo momento in modo da non doverla scrivere tutta in poco tempo alla fine.

1.7 Analisi dei mezzi

Elencare e descrivere i mezzi disponibili per la realizzazione del progetto. Ricordarsi di sempre descrivere nel dettaglio le versioni e il modello di riferimento.

1.7.1 Software

Programmi installati:

- PuTTY versione 0.70

Librerie di codice utilizzate:

- Bootstrap 4.3.1
- PHPMailer 6.0.7
- jQuery 3.3.1
- Notify.js 2015
- JUnit Jupiter 5.0-M1

1.7.2 Hardware

Macchina server:

- Ubuntu Server 18.4
- 1 GB di memoria RAM
- 25 GB di disco disponibili

Gestione casinò

2 Progettazione

2.1 Design dell'architettura del sistema

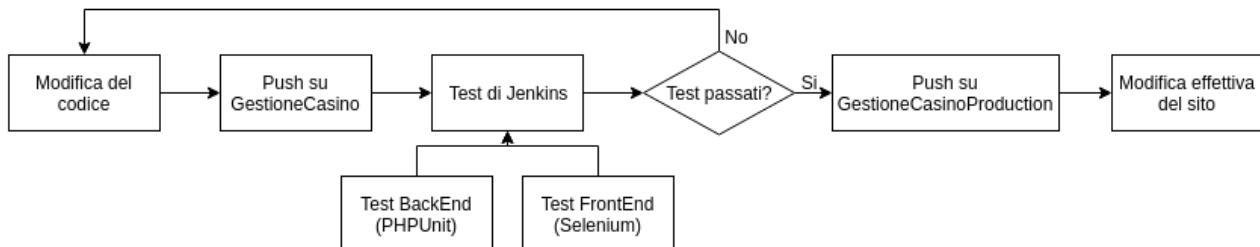


Figura 7 - Design dell'architettura del sistema

2.2 Design dei dati e database

Il database sviluppato per essere utilizzato con questa applicazione è stato progettato tramite il seguente schema E/R e logico:

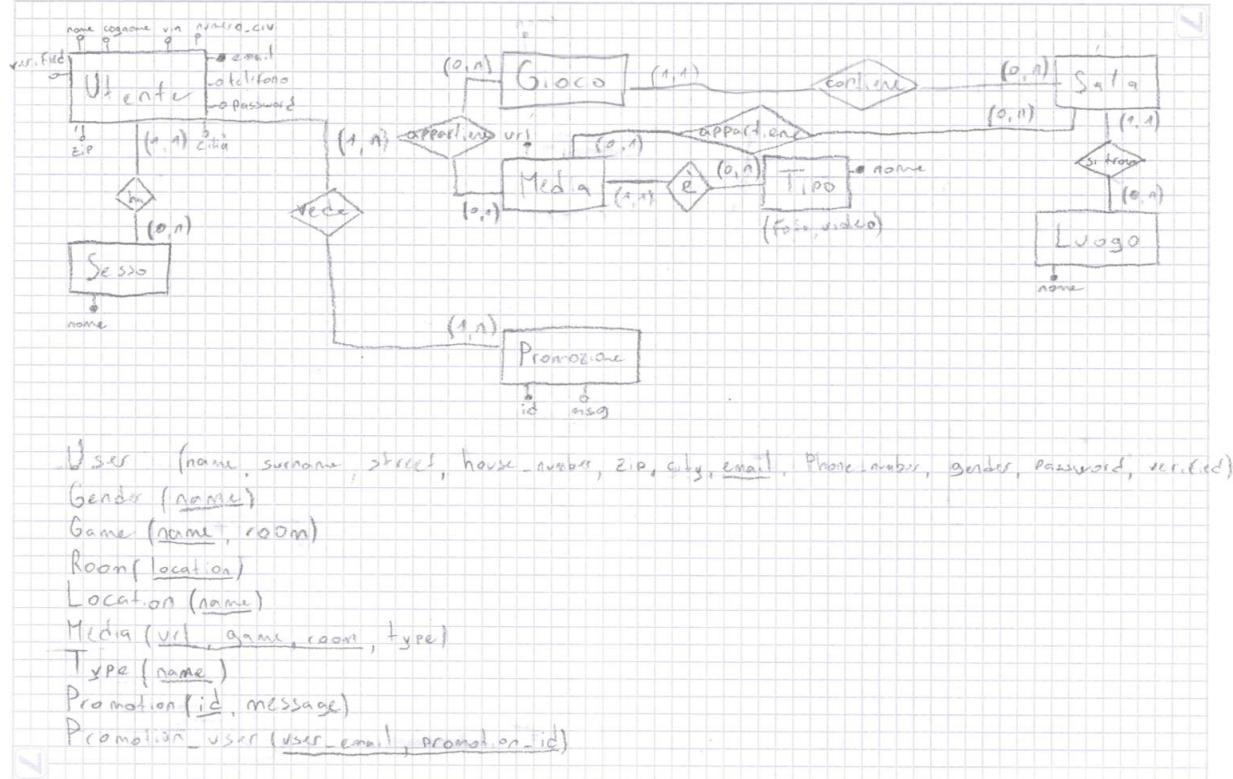


Figura 8 - Diagramma E/R usato per la progettazione del database

Come prima cosa, a sinistra troviamo la tabella dell'**Utente**, che viene identificata da nome, cognome, sesso (i cui valori predefiniti sono "Male" e "Female" contenuti nella tabella **Gender**), via, numero civico, NAP, città, indirizzo e-mail, numero di telefono, password e un valore booleano che dice se è stato verificato o meno. In seguito abbiamo la tabella **Gioco**, che contiene un nome e un riferimento a una sala. Poi troviamo **Sala**, identificata unicamente da un luogo, ossia un valore predefinito contenuto nella tabella **Location**. Andando avanti c'è **Media**, ossia la tabella che contiene i riferimenti alle immagini e i video associati ai giochi o alle sale, che contiene l'URL che punta a quest'immagine o video, il gioco o sala a cui è associato e il tipo di file multimediale di cui si tratta, appunto video, immagine, ... Questi sono valori specificati all'interno della tabella **Type**. Dopodiché c'è **Promozione**, identificata da un codice identificativo e un messaggio da

mostrare agli utenti specificati all'interno della tabella **Promotion_user**, che definisce appunto a quale utente deve venire mostrato quale promozione.

La progettazione è però stata modificata in seguito, ma i cambiamenti erano talmente piccoli che non abbiamo ritenuto necessario un nuovo diagramma E/R. I cambiamenti sono i seguenti:

- Aggiunta la colonna "type" alla tabella "user". Essa contiene il tipo di utente in modo da potergli mostrare o meno una determinata promozione.
- Modificata la tabella "media" in modo che non contenga riferimenti (foreign key) ad altre tabelle, e sono state aggiunte tre tabelle ponte al loro posto, una tra "game" e "media", "game_media", una tra "promotion" e "media", "promotion_media", e una tra "room" e "media", "room_media".
- Aggiunta una tabella che contiene i valori predefiniti per i tipi di utente chiamata "user_type"
- Rinominata la tabella "type" in "media_type" per non confonderla con "user_type"

Il risultato di questa progettazione è il seguente database:

- Game (room, name, description)
- Game_media (game_name, media_url)
- Gender (name)
- Media (url, type)
- Media_type (name)
- Promotion (id, name, description)
- Promotion_media (promotion_id, media_url)
- Promotion_user (user_type, promotion_id)
- Room (location, description)
- Room_media (room_location, media_url)
- User (name, surname, street, house_number, zip_code, city, email, phone_number, gender, password, type, verified, admin)
- User_type (name)

2.3 Design procedurale

Le classi presenti in questo progetto sono le seguenti, divise per linguaggio:

2.3.1 PHP

2.3.1.1 User

La classe sulla quale si basano i test relativi agli utenti.

User
<pre>- \$name - \$surname - \$birthday - \$city - \$zipCode - \$houseNumber - \$telephoneNumber - \$email - \$gender - \$password + __construct(\$name, \$surname, \$birthday, \$city, \$zipCode, \$houseNumber, \$telephoneNumber, \$email, \$gender, \$password) + tryEmail(\$email) + tryName(\$object) + tryDate(\$object) + getAge(\$date) + tryNumber(\$object) + tryHouseNumber(\$object) + tryGender(\$object) + tryPassword(\$object) + tryZipCode(\$object) + getName() + getSurname() + getBirthday() + getCity() + getZipCode() + getAddress() + getHouseNumber() + getTelephoneNumber() + getEmail() + getGender() + getPassword()</pre>

Figura 9 - Diagramma UML utilizzato per la progettazione della classe User

2.3.1.2 DatabaseTestCase

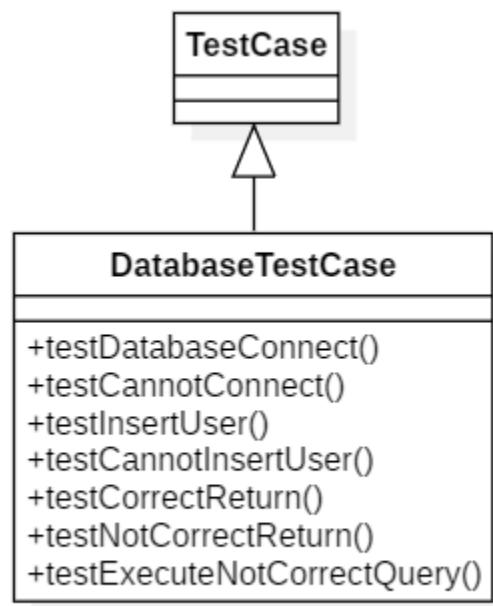


Figura 10 - Diagramma UML utilizzato per la progettazione della classe DatabaseTestCase

2.3.1.3 SendMailTest

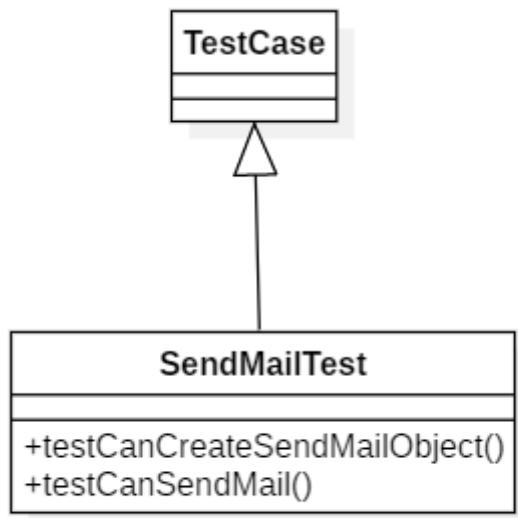


Figura 11 - Diagramma UML utilizzato per la progettazione della classe SendMailTest

2.3.1.4 UserTestCase

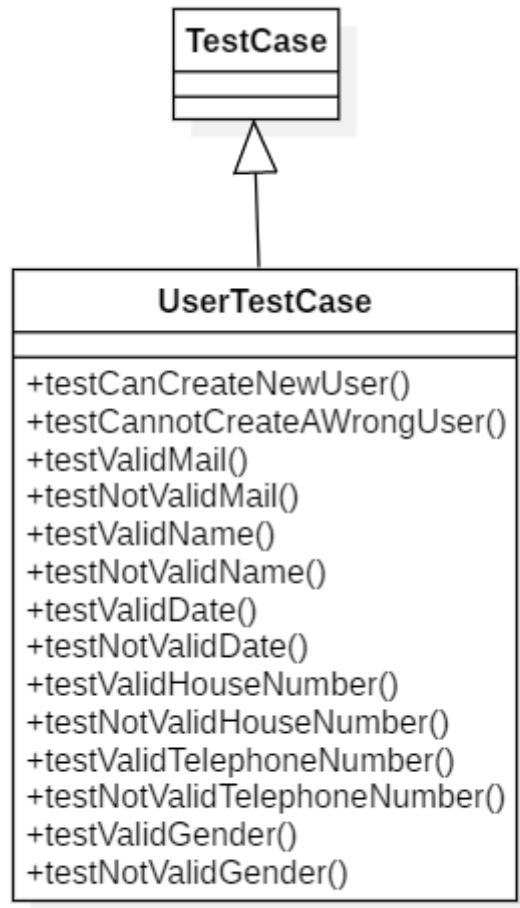


Figura 12 - Diagramma UML utilizzato per la progettazione della classe UserTestCase

2.3.2 Java

2.3.2.1 LoginTest

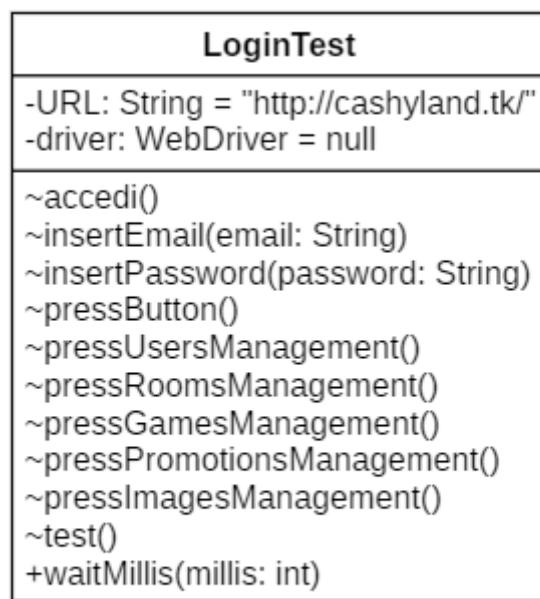


Figura 13 - Diagramma UML utilizzato per la progettazione della classe LoginTest

2.3.2.2 NavigationTest

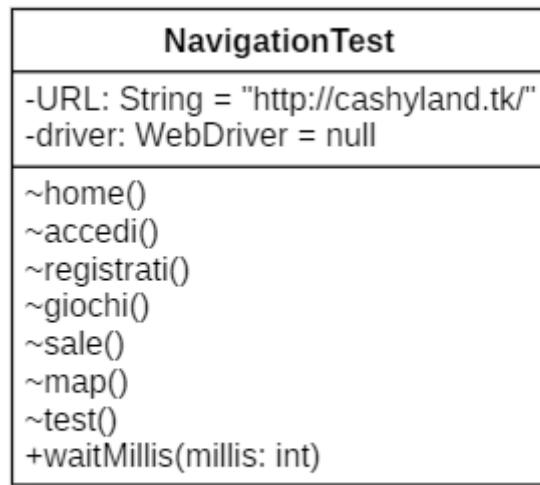


Figura 14 - Diagramma UML utilizzato per la progettazione della classe NavigationTest

2.3.2.3 RegistrationTest

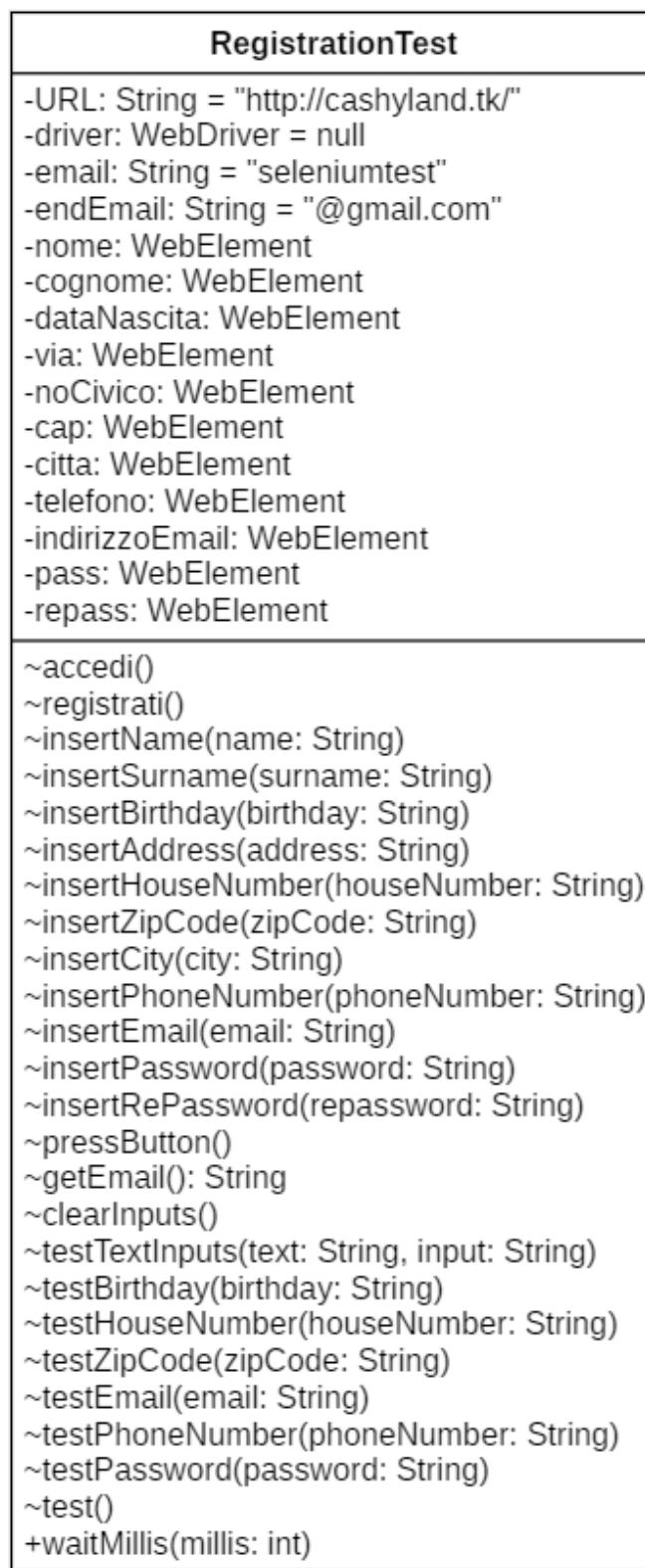


Figura 15 - Diagramma UML utilizzato per la progettazione della classe RegistrationTest

2.3.2.4 UserUpdateTest

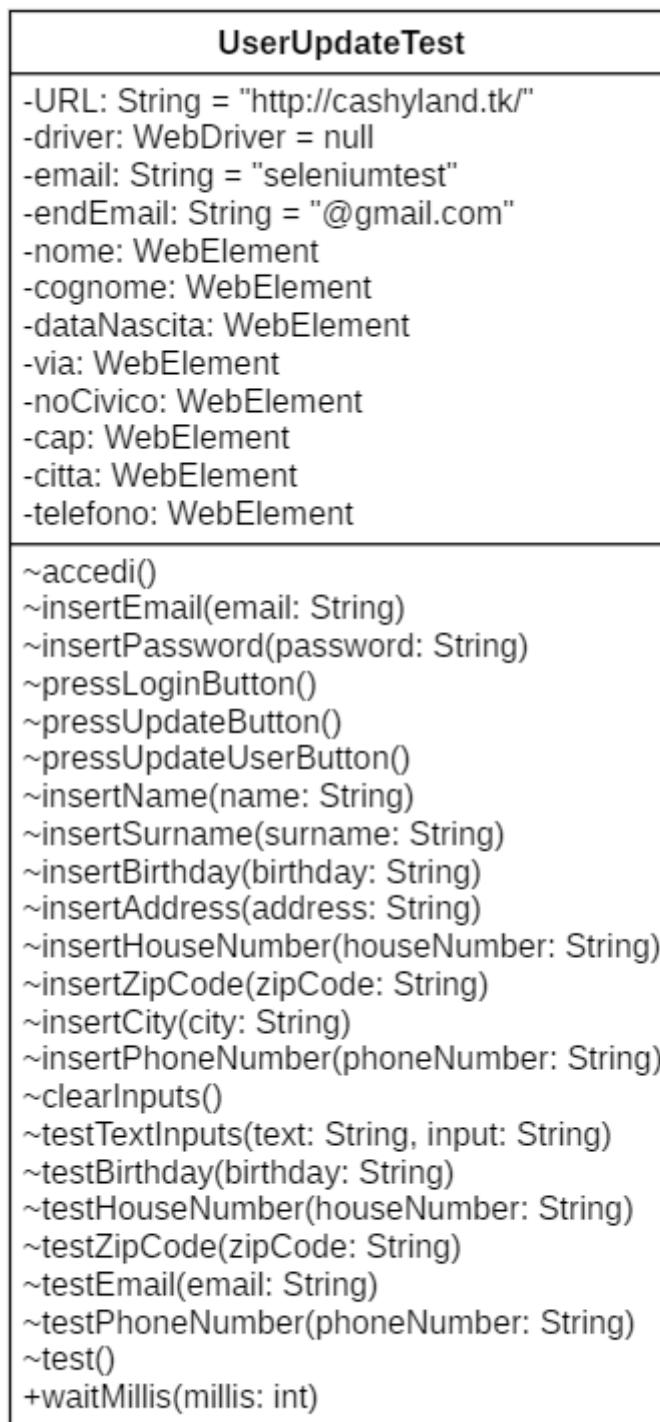


Figura 16 - Diagramma UML utilizzato per la progettazione della classe UserUpdateTest

3 Implementazione

3.1 Front-end (pagine web – HTML + CSS)

La pagina iniziale alla quale l'utente finale si collega è la pagina home. In questa pagina viene descritto brevemente come è composto il casinò, in altre parole viene mostrato l'edificio accompagnato da una piccola descrizione e un pulsante che, in caso dovesse essere cliccato, porterebbe alla pagina di Google Maps contenente la posizione della struttura. Proseguendo per la pagina si può trovare una piccola presentazione sui giochi offerti accompagnata da un'immagine e, anche qui, un bottone che, se cliccato, porta alla pagina dei giochi. La pagina principale di presenta così:

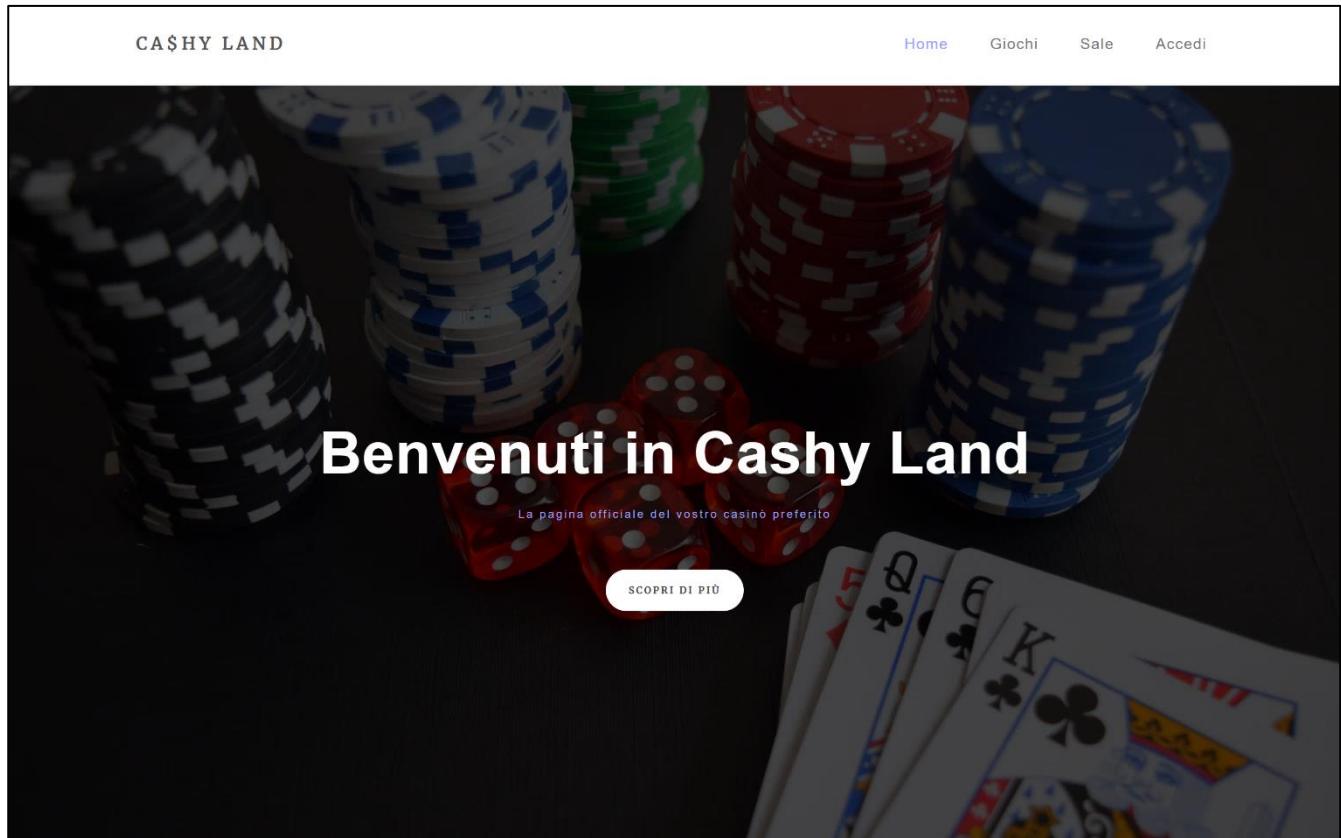
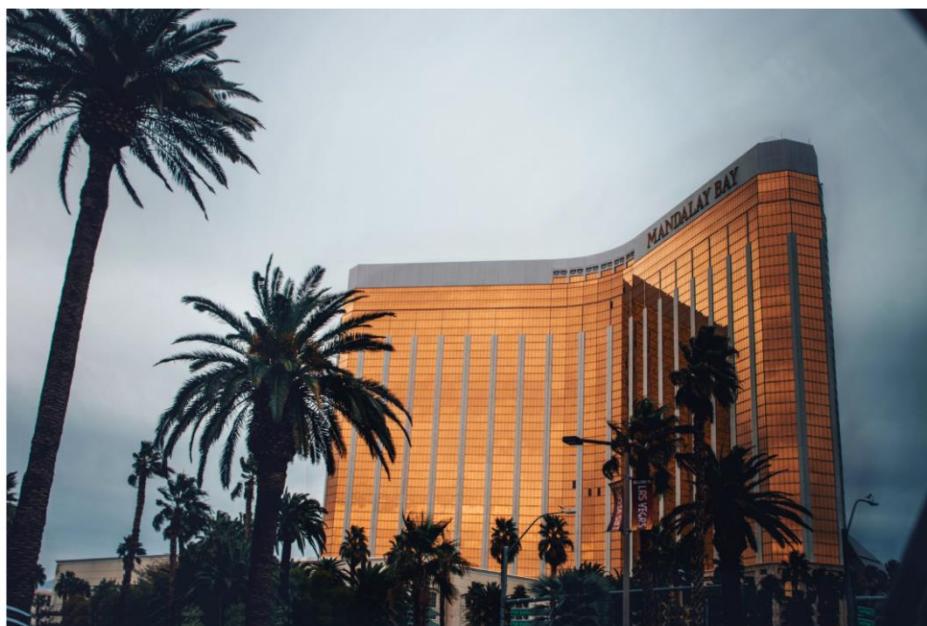


Figura 17 - Pagina di benvenuto 1

Questa pagina è composta solo da parti grafiche (html, css e javascript per le animazioni) e link, quindi non viene elaborato alcun tipo di dato. Nella navbar (barra di navigazione) in alto sono presenti tutti i link che portano alle pagine visitabili all'interno del sito: il link "Home" è la pagina stessa, il link "Giochi" porterà alla pagina dei giochi che il casinò offre, il link "Sale" porterà alla pagina delle sale del casinò e infine il link "Accedi" porterà alla pagina di login alla quale l'utente può accedere o eventualmente registrarsi per ricevere delle promozioni utilizzabili all'interno della struttura. Come in ogni altra pagina, è presente anche il footer (piè di pagina) dove vengono mostrate eventuali informazioni per poterci contattare.

Proseguendo ci sono la pagina dei giochi, la pagina delle sale e la pagina delle promozioni. Queste tre pagine sono effettuate tutte allo stesso modo per quanto riguarda la parte estetica, infatti all'interno di queste pagine ogni gioco/sala/promozione ha un titolo, un'immagine e una breve descrizione. Questa pagina ha la seguente struttura:



[Titolo]

[Descrizione]

Figura 18 - Pagina di benvenuto 2

Il template utilizzato per ogni “oggetto” della pagina è una sezione con all'interno un div il quale, a sua volta, viene riempito con un'immagine, un titolo e una descrizione, come il codice sottostante:

```
<section class="blog">
    <div class="container">
        <div class="row">
            <div class="col-md-offset-1 col-md-10 col-sm-12">
                <div class="blog-post-thumb">
                    <div class="blog-post-image" id="[Num]">
                        
                    </div>
                    <div class="blog-post-title">
                        <h3>[Titolo]</h3>
                    </div>
                    <div class="blog-post-des">
                        <p>[Descrizione]</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
```

Ritornando alla navbar descritta precedentemente cliccando il link “Accedi” l’utente viene portato alla pagina di login.

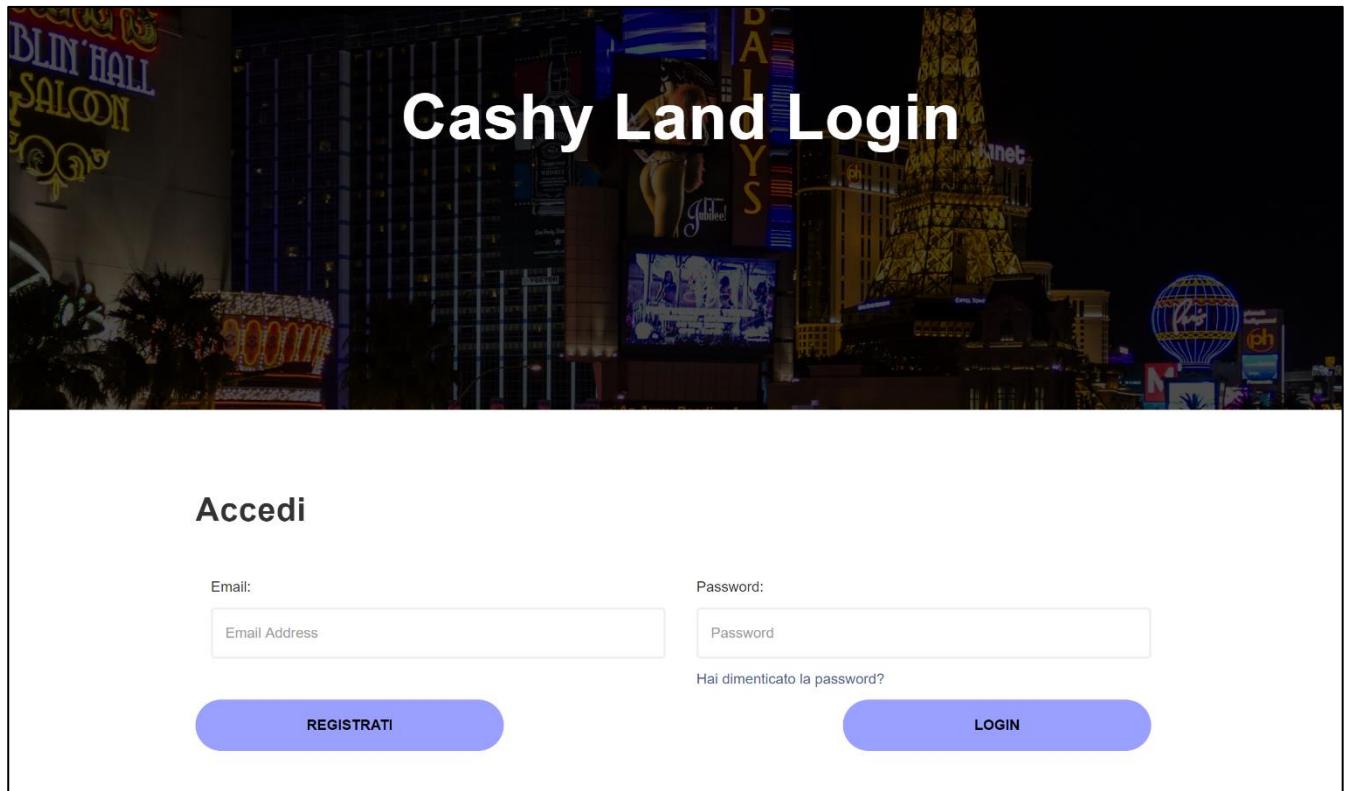


Figura 19 - Pagina di accesso 1

In questa pagina l’utente ha la possibilità di accedere al nostro sito oppure, se quest’ultimo non avesse un account proprio, la possibilità di registrarsi. Inoltre offre l’opportunità, in caso venisse dimenticata la password, di poterla modificare. Per fare ciò bisogna cliccare sulla domanda “Hai dimenticato la password?” e l’utente verrà portato sulla pagina di recupero.

In questa pagina l'utente dovrà inserire la propria mail utilizzata per la registrazione cosicché il server possa inviare a quest'ultimo una mail di recupero. Questa pagina è stata creata come l'immagine seguente:

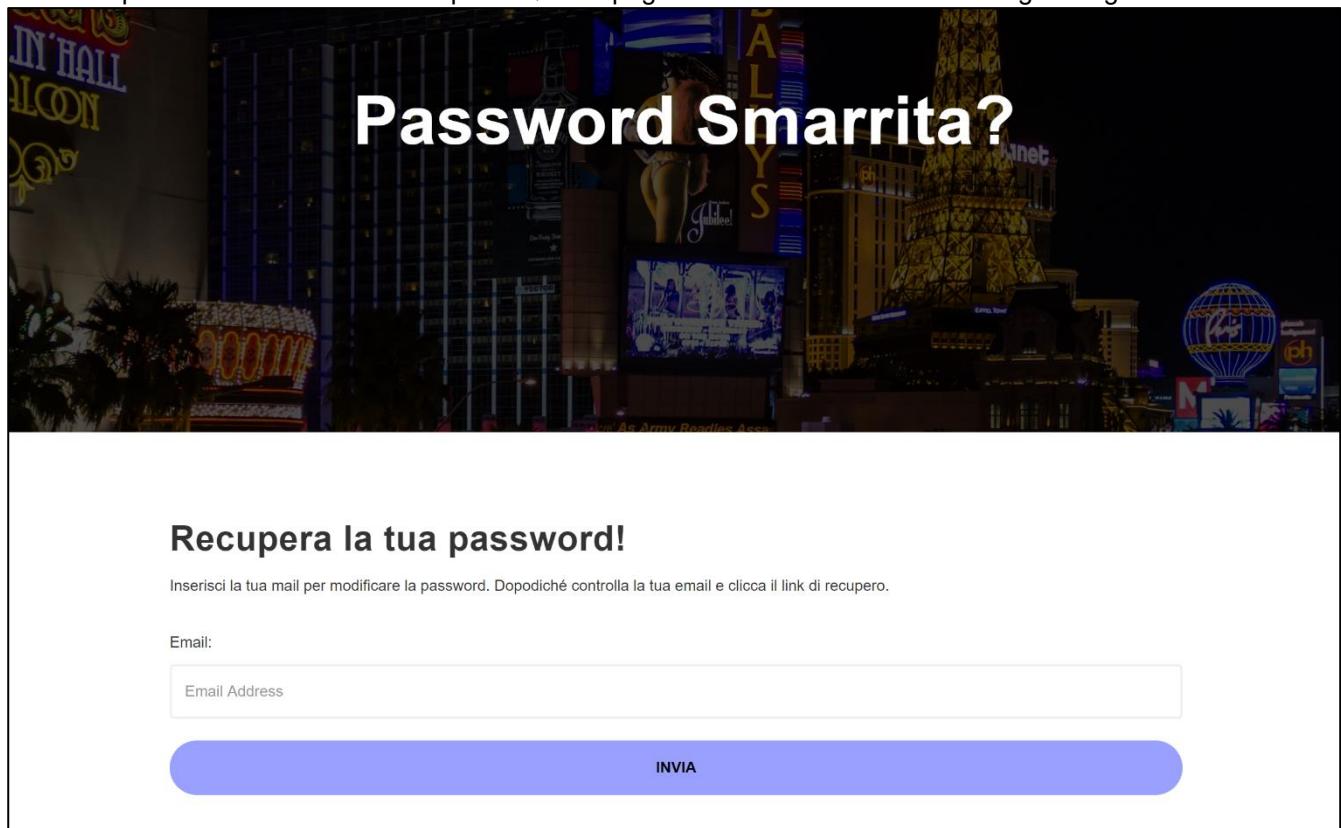


Figura 20 - Pagina di recupero password

Ritornando alla pagina di login, quando l'utente vuole accedere avviene un semplice controllo sui dati inseriti lato server, in caso venisse inserita una mail inesistente o una password non valida, il server invia un cookie al client e quest'ultimo mostra un messaggio di errore all'utente finale. Per il controllo lato server verrà dedicata una descrizione più avanti, invece per mostrare il messaggio di errore viene utilizzata la libreria notify.js. Il codice da utilizzare per mostrare una notifica è il seguente:

```
$.notify(<messaggio>, { position:<posizione>});
```

Per avere un risultato simile:

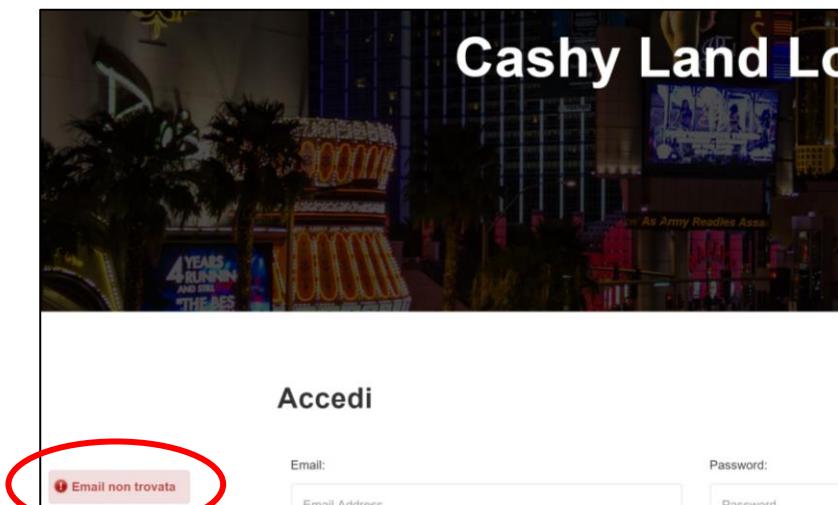


Figura 21 - Pagina di accesso 2

Invece se l'utente volesse registrarsi alla nostra pagina è sufficiente cliccare il pulsante “REGISTRATI” nella pagina di login. A questo punto l'utente verrà portato nella pagina di registrazione.



Registrazione

Accedi

Nome:	Cognome:
<input type="text" value="Nome"/>	<input type="text" value="Cognome"/>
Data Nascita:	Via:
<input type="text" value="gg/mm/aaaa"/>	<input type="text" value="Via"/>
No. Civico:	CAP:
<input type="text" value="Numero Civico"/>	<input type="text" value="CAP"/>
Città:	No. Telefono:
<input type="text" value="Città"/>	<input type="text" value="Numero di Telefono"/>
Sesso:	Email:
<input type="text" value="Maschio"/>	<input type="text" value="Email"/>
Password:	Ripeti Password:
<input type="text" value="Password"/>	<input type="text" value="Ripeti Password"/>

REGISTRATI

Figura 22 - Pagina di registrazione 1

La pagina di registrazione è formata da un form in HTML costituita da 12 campi input. Tutti campi sono obbligatori, quindi se uno di quest'ultimi non viene compilato verrà mostrata sullo schermo una notifica di errore mostrando come bisogna compilarlo. Inoltre il campo input errato verrà contrassegnato con uno sfondo di colore rosso. Il controllo della validazione dei campi viene gestito in due parti: una lato client e l'altra lato server. Il controllo lato server verrà spiegato successivamente, per quanto riguarda il controllo lato client è stato svolto in JavaScript. Un esempio di codice di controllo è questo (controllo della data di nascita):

```
//Controllo della data di nascita
function checkDate(val){
    var dataN = val.split("-");
    var data_nascita = dataN[0] + "/" + dataN[1] + "/" + dataN[2];
    data_nascita = new Date(data_nascita);
    var dataMassima = new Date();
    dataMassima.setFullYear(dataMassima.getFullYear() - 100);
    var dataMinima = new Date();
    dataMinima.setFullYear(dataMinima.getFullYear() - 18);
    return (!(data_nascita > dataMinima) || (data_nascita < dataMassima) || dataN ==
    ""));
}
```

In caso che il dato inserito dall'utente finale fosse errato verrà mostrato a schermo questa serie di fatti:

The screenshot shows a registration form with various fields. The 'Data Nascita' field (Birth Date) contains '08/05/2019' and is highlighted with a red oval, indicating it is the current focus of validation. Below the form, a message 'Data non valida!' (Invalid date!) is displayed in a red box, also circled in red. Other fields visible include 'Via:' (via ro, casa), 'No. Civico:' (0000), 'CAP:' (0000), 'Città:' (Città), 'No. Telefono:' (0000000000), 'Sesso:' (Maschio), 'Email:' (mail@mail.mail), 'Password:' (*****), and 'Ripeti Password:' (*****). A blue 'REGISTRATI' button is at the bottom right.

Figura 23 - Pagina di registrazione 2

Invece per sapere come deve venir compilato un client viene mostrata una notifica sopra all'input nel seguente modo:

Cognome:

Cognome

● Max 50 caratteri, solo lettere e caratteri da scrittura

Figura 24 - Notifica campi pagina registrazione

Una volta effettuata la registrazione nel modo corretto (cioè senza errori) verrà richiesto di verificare la mail per far sì che all'interno del database non ci siano persone "robot" (finte) così da occupare meno spazio nella memoria. Dunque verrà mostrata a schermo questa pagina:

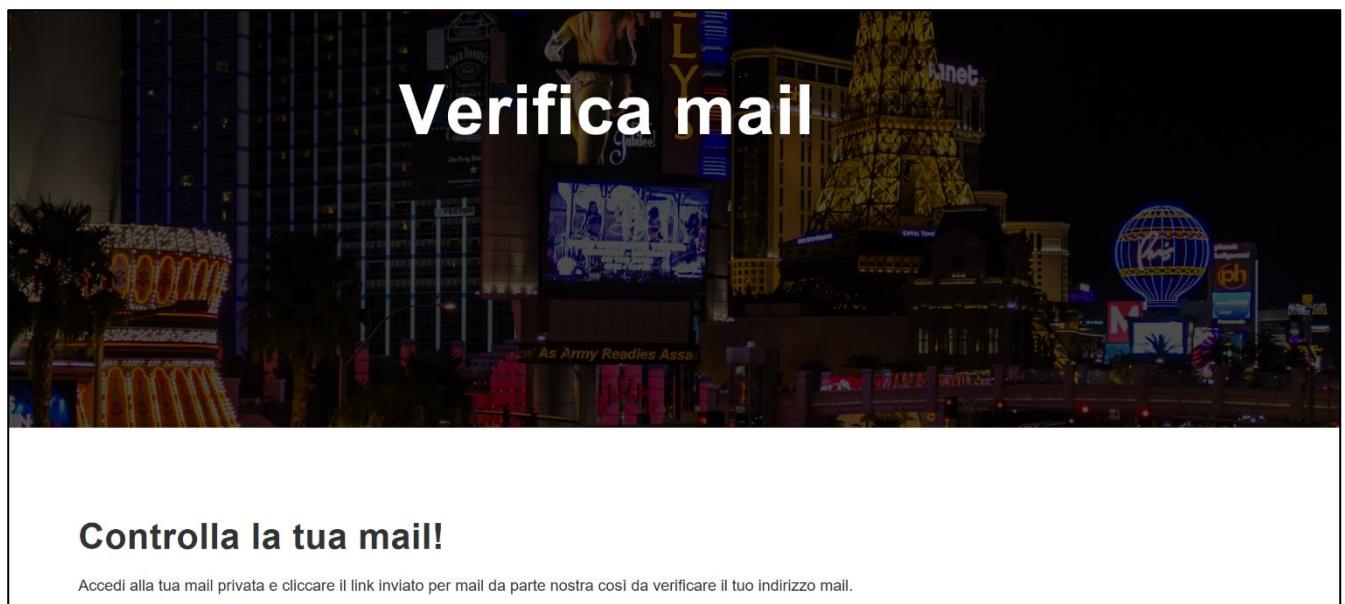


Figura 25 - Registrazione completata

A questo punto verrà ricevuta una mail e l'utente dovrà accedere alla propria mail privata e cliccare il link presente all'interno del messaggio. Adesso potrà accedere ad una pagina riservata al suo profilo. In questa pagina si potranno modificare i propri dati ed eventualmente la propria password.

La pagina si adatta in modo dinamico al tipo di persona che avrà fatto l'accesso. Il tipo di persona si differenzia tra admin e utente; quando si effettuerà l'accesso come utente si avrà l'opzione aggiuntiva di visualizzare le promozioni riguardanti al casinò, cioè la possibilità di accedere alla pagina dove vengono mostrati tutti i buoni (come già detto prima, ha la stessa struttura delle pagine sale e giochi). Invece, in caso che la persona che effettua l'accesso sia un admin, la pagina si mostra diversamente, infatti non verrà più mostrata l'opzione di visualizzare le promozioni ma una serie di gestioni per le pagine giochi, sale e promozioni e per gli utenti. L'admin attraverso queste pagine potrà aggiungere, modificare o rimuovere qualsiasi tipo di cosa.

La pagina per un utente normale si presenta così:

Bentornato Utente

Informazioni Base:

Nome: Utente
Cognome: Prova
Nascita: 0000-00-00

[Modifica Dati](#)

[Modifica password](#)

Modifica Password:

Email: prova@yopmail.com

Visualizza Promozioni:

Cliccando il bottone qui sotto puoi visualizzare tutte le promozioni da utilizzare all'interno del nostro casinò:

[Visualizza Promozioni](#)

Figura 26 - Pagina di profilo 1

Cliccando sul bottone “Modifica Dati”, come già detto precedentemente, si potranno modificare i propri dati, come nome, cognome, data di nascita, ecc. La pagina di modifica dei dati è molto simile alla pagina di registrazione, infatti verranno nuovamente controllati i dati, però vengono rimossi il campo della password e della e-mail.

Cliccando il tasto “Visualizza Promozioni” si potranno visualizzare le promozioni utilizzabili all'interno dell'edificio.

Infine cliccando sul bottone “Modifica Password”, si potrà modificare la password dopo una conferma per e-mail di richiesta di cambiamento password.

Una volta cliccata il link di recupero attraverso la mail ricevuta l'utente verrà mandato ad una pagina di reset password:

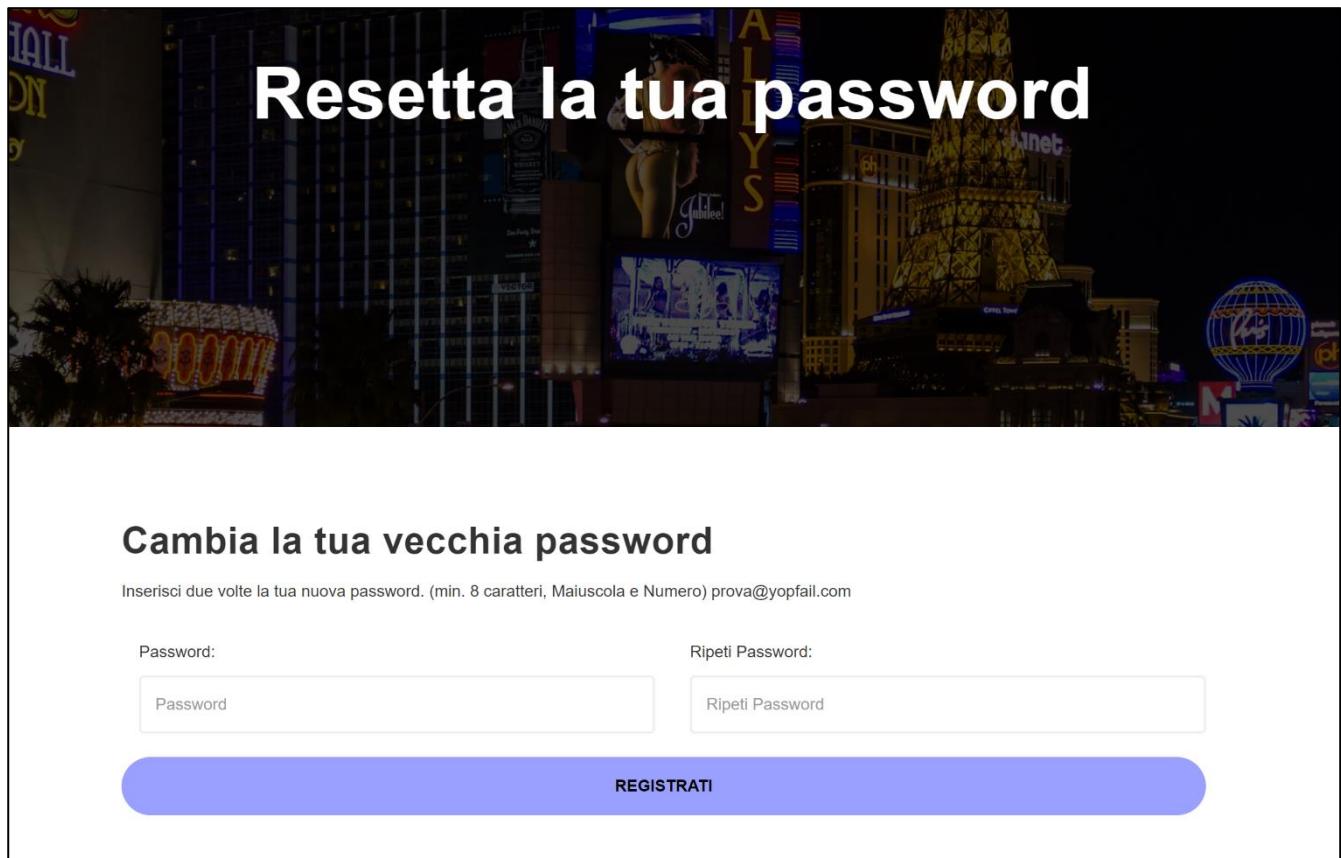


Figura 27 - Pagina di recupero password 2

In questa pagina verrà richiesto di inserire una nuova password sempre con le stesse regole, cioè minimo 8 caratteri, almeno una maiuscola e almeno un numero. Un ulteriore controllo in JavaScript è quello si controllare che le due Password siano uguali. Il codice è il seguente:

```
//Controllo se le password sono uguali
function checkPassword(pas1, pas2){
    return (pas1 == pas2) && pas2.toLowerCase() != pas1 && /\d/.test(pas1) && pas1.length > 7;
}

//Controllo lato client dei campi con eventuali notifiche in caso di errore.
function checkAll(){
    var inputs = document.getElementsByTagName("input");
    if(checkPassword(inputs[0].value, inputs[1].value)){
        document.getElementById("registration_form").submit();
    }else{
        if(!checkPassword(inputs[0].value, inputs[1].value)){
            inputs[0].style.backgroundColor = "#ffcccc";
            inputs[1].style.backgroundColor = "#ffcccc";
        }
    }
}
```

```
    $.notify("Password non valida o non corrispondente!", { position:"bottom left" });
}else{
    inputs[0].style.backgroundColor = "white";
    inputs[1].style.backgroundColor = "white";
}
}
```

Invece per un admin vengono aggiunte, come già detto prima, le gestioni di per le varie pagine e la gestione per gli utenti, rimuovendo la possibilità di visualizzare le promozioni. La pagina offrirà dunque le seguenti quattro opzioni:

Gestione Utenti:

Puoi gestire tutti gli utenti del sito, aggiungere o togliere qualsiasi utente:

[Modifica Utenti](#)

Gestione Sale:

Puoi gestire tutte le sale del sito, aggiungere o togliere qualsiasi sala:

[Modifica Sale](#)

Gestione Giochi:

Puoi gestire tutti i giochi del sito, aggiungere o togliere qualsiasi gioco:

[Modifica Giochi](#)

Gestione Promozioni:

Puoi gestire tutte le promozioni del sito, aggiungere o togliere qualsiasi promozione:

[Modifica Promozioni](#)

Figura 28 - Pagina di profilo amministratore

La visualizzazione dinamica della pagina è stata fatta con un semplice “if - else if” in php; tramite la variabile \$queryRepose[“admin”] si può sapere se la persona che ha eseguito l’accesso è un amministratore o un semplice utente, attraverso questo codice:

```
<?php
if($queryRepose["admin"]==0){ //Entra se è un utente normale
    echo '...';
}else if($queryRepose["admin"]==1){ //Entra se è un admin
    echo '...';
}
?>
```

Le pagine di gestione sono molto simili fra loro, permettono di aggiungere, modificare o rimuovere utenti, sale, giochi e promozioni. Prendiamo per esempio la gestione delle sale quindi clicchiamo “Modifica Sale”.

Inserire dati

location:

description:

AGGIUNGI

Figura 29 - Pagina di gestione sala 1

Questa è la prima cosa che ci si trova davanti, un form contenente due input e un pulsante. Il primo input “location” indica il titolo (in questo caso il titolo della sala, che può inerire al piano oppure alla posizione di dove questa si trovi), il secondo input “description” indica la descrizione della sala. Una volta riempiti questi due input si può procedere al bottone “AGGIUNGI” e quest’ultimo aggiungerà la sala al database e a sua volta alla pagina delle sale. Questo vale allo stesso modo per la gestione dei giochi e delle promozioni.

Vedi già i dati presenti:

Location	Description
Primo Piano	In questa sala si possono trovare diversi tavoli da gioco per il Black Jack. Non resta altro che andare a provare a vincere!
Secondo Piano	In questa sala potrete divertirvi a giocare alle nostre infinite slot machines. La vincita è garantita!

Figura 30 - Pagina di gestione sala 2

Scorrendo in giù per la pagina troviamo questa tabella che permette la modifica di una sala già creata (cliccando sulla matitina) oppure l’eliminazione di quest’ultima cliccando sul cestino. La tabella è stata resa responsive tramite le classi di bootstrap come dice il seguente codice:

```
<div class="col-md-offset-1 col-md-10 col-sm-12 table-responsive text-nowrap">
  <table class='table' style='table table-striped'>
    [Tabella]
  </table>
</div>
```

3.2 Back-end (PHP)

3.2.1 Classe user:

3.2.1.1 Costruttore:

Per la gestione degli utenti lato php è stata creata una classe “User” che si occupa di verificare le varie informazioni legate all’utente. Il codice riportato sopra è la parte più importante della classe “User”. Come spiegato prima il costruttore da parametro riceve tutte le informazioni che gli servono per instanziare un nuovo utente. La classe è suddivisa in una parte a oggetti e una parte statica. Tutti i metodi che caratterizzano un utente sono a oggetti mentre quelli legati ai test sono statici, in seguito verrà mostrato un esempio.

```
private $name;
private $surname;
private $birthday;
private $city;
private $address;
private $houseNumber;
private $telephoneNumber;
private $email;
private $gender;
private $password;
private $zipCode;

/**
 * Costruttore personalizzato che si occupa di inizializzare gli attributi di un utente.
 * Prima di inizializzare controlla che l'informazione sia corretta.
 */
public function __construct($name,$surname,$birthday,$city,$zipCode,$address,$houseNumber,$telephoneNumber,$email,$gender,$password)
{
    $this->name = User::tryName($name);
    $this->surname = User::tryName($surname);
    $this->birthday = User::tryDate($birthday);
    $this->city = User::tryName($city);
    $this->address = User::tryName($address);
    $this->houseNumber = User::tryHouseNumber($houseNumber);
    $this->telephoneNumber = User::tryNumber($telephoneNumber);
    $this->email = User::tryEmail($email);
    $this->gender = User::tryGender($gender);
    $this->zipCode = User::tryZipCode($zipCode);
    $this->password = User::tryPassword($password);
}
```

3.2.1.2 Metodo di test:

Il seguente codice è un esempio per mostrare come funzionano i metodi di test. Come specificato prima il metodo è statico quindi può essere chiamato ovunque senza dover instanziare un nuovo *User*. Viene eseguito il codice di test, in questo caso viene verificato che sia più lungo di 8 caratteri e che ci sia almeno una lettera maiuscola. In seguito se il test è andato a buon fine allora si ritorna l’oggetto passato, altrimenti lancia una eccezione con un messaggio personalizzato.

```
public static function tryPassword($object){
    if(strlen($object)>=8){
        if(strtowlower($object) != $object){
            return $object;
        }else{
            throw new InvalidArgumentException(
                sprintf( "%s have all lower character", $object)
            );
        }
    }
}
```

```
    }else{
        throw new InvalidArgumentException(sprintf( '"%s" is too short', $object));
    }
}
```

3.2.2 Classe Database:

3.2.2.1 Costruttore:

La classe per la connessione al database è composta da parecchi tool che possono essere utili. Prima di tutto nel costruttore della classe riceve tutte le informazioni necessarie per il giusto collegamento al database, come (nome del database, ip, porta, username, password). E istanzio una nuova connessione.

```
private $db;

function __construct($host,$port,$dbname,$username,$password)
{
    $this->db = new PDO(
        "mysql:host=$host;port=$port;dbname=$dbname", $username, $password
    );
}
```

3.2.2.2 Metodo per esecuzione query:

Questo metodo si occupa di eseguire e ritornare il risultato di una query. Il risultato che torna è già fetchato, oppure può anche non esserlo, quindi pronto all'uso. Nel caso la query che si sta andando a fare è sbagliata istanzio e lancio un'eccezione con messaggio personalizzato.

```
public function executeQuery($query) {
    $result = $this->db->query($query);
    if ($result === FALSE) {
        throw new InvalidArgumentException(
            "Failed to load schema is not exists or you are not permission");
    }
    return $result->fetchAll();
}

public function executeQueryWithoutFetch($query) {
    $result = $this->db->query($query);
    if ($result === FALSE) {
        throw new InvalidArgumentException(
            "Failed to load schema is not exists or you are not permission");
    }
    return $result;
}
```

3.2.2.3 Metodo per stampare una tabella:

Una dei tool citati in precedenza è la possibilità di stampare una tabella già formattata per gli standard bootstrap. Il seguente metodo si occupa di stampare una tabella in base alla query che riceve.

```
public function printTableQuery($selectQuery) {
    $result = $this->db->query($selectQuery);
    if ($result === FALSE) {
        throw new InvalidArgumentException(
            "Failed to load schema is not exists or you are not permission");
    }
    $result = $result->fetchAll();
    echo "<table class='table' style='overflow-x:auto;'><thead><tr>";
    echo "<th></th>";
    echo "<th></th>";
    $n = 0;
    foreach ($result[0] as $key => $value) {
        if($n%2==0){
            echo "<th><b>".strtoupper($key{0}).substr($key,1,strlen($key))."</b></th>";
        }
        $n++;
    }
    echo "</tr></thead><tbody>";
    for ($i=0; $i < sizeof($result); $i++) {
        echo "<tr>";
        $n = 0;
        foreach ($result[$i] as $key => $value) {
            if($n%2==0){
                echo "<td><b>".strtoupper($key{0}).substr($key,1,strlen($key))."</b></td>";
            }
            $n++;
        }
        echo "</tr>";
    }
}
```

Gestione casinò

```
echo "<th><a href='php/database/modify.php?value=modify_&i'><i class='fa fa-pencil' id='modify_&i'></a></th>";
echo "<th><a href='php/database/modify.php?value=delete_&i'><i class='fa fa-trash' id='delete_&i'></a></th>";
for ($j=0; $j < sizeof($result[$i])/2; $j++) {

    //echo "<tr><i class='far fa-trash-alt' id='&i'></tr>";
    echo "<th>".$result[$i][$j]."</th>";
}
echo "</tr>";
}
echo "</tbody></table>";
```

3.2.2.4 Metodo per aggiungere un utente:

Questo metodo tramite un parametro di tipo “User” si occupa di inserire nel database un utente e inviar gli già la mail per la verifica.

```
public function insertUser($user){
    if(gettype($user) == "object"){
        if(get_class($user) == "User"){
            $name = $user->getName();
            $birthday = $user->getBirthday();
            $surname=$user->getSurname();
            $street=$user->getAddress();
            $house_number=$user->getHouseNumber();
            $zip_code=$user->getZipCode();
            $city=$user->getCity();
            $email=$user->getEmail();
            $phone_number=$user->getTelephoneNumber();
            $gender=$user->getGender();
            $password=$user->getPassword();

            $query = "Insert into user
(
    name,
    surname,
    street,
    house number,
    zip_code,
    city,
    email,
    phone_number,
    gender,
    password,
    verified,
    birthday,
    type,
    admin
)
values(
    '$name',
    '$surname',
    '$street',
    $house_number,
    $zip_code,
    '$city',
    '$email',
    $phone_number,
    '$gender',
    '$password',
    0,
    $birthday,
    'occasionale',
    0
)";
            $this->executeQuery($query);
        }else{
            throw new InvalidArgumentException(get_class($user)." is not a User class");
        }
    }else{
        throw new InvalidArgumentException(gettype($user)." is not a User class");
    }
}
```

}

3.2.3 Classe SendMail:

3.2.3.1 Costruttore:

Il costruttore di questa classe si occupa di stabilire un connessione con il server che fa da sender delle mail. Nel nostro caso abbiamo usato una libreria esterna per fare questo ovvero “*PHPMailer*” con relativa documentazione su github (<https://github.com/PHPMailer/PHPMailer/blob/master/README.md>).

Inizialmente avevamo pensato di usare hotmail come server smtp, però dopo qualche mese di utilizzo abbiamo riscontrato troppi errori. Nella maggior parte delle volte hotmail bannava l’account e lo rendeva inutilizzabile e bisognava sbloccarlo a mano ogni volta. Abbiamo quindi deciso di passare a un account gmail, che personalmente trovo più efficiente.

```
public function __construct()
{
    $this->mail = new PHPMailer(true);
    $this->mail->isSMTP();
    $this->mail->Host = 'smtp.gmail.com';
    $this->mail->SMTPAuth = true;
    $this->mail->Username = 'gruppocasino2018@gmail.com';
    $this->mail->Password = 'Casin02018';
    $this->mail->SMTPSecure = 'tls';
    $this->mail->Port = 25;
    $this->mail->From = 'gruppocasino2018@gmail.com';
    $this->mail->FromName = 'Verify Password';
}
```

3.2.3.2 Metodo mailSend:

Questo metodo si occupa semplicemente di spedire il messaggio, in base ai parametri ricevuti.

```
public function mailSend($email,$subject,$message) {
    $this->mail->addAddress($email);
    $this->mail->isHTML(true);
    $this->mail->Subject = $subject;
    $this->mail->Body = $message;
    return $this->mail->send();
}
```

3.2.4 Esempio utilizzo classi:

3.2.4.1 Registrazione utente

```
$u = new User();

        $_POST["firstname"],
        $_POST["surname"],
        $_POST["birthday"],
        $_POST["city"],
        $_POST["zipCode"],
        $_POST["address"],
        $_POST["houseNumber"],
        $_POST["phoneNumber"],
        $_POST["email"],
        $_POST["gender"],
        $_POST["password"],
        $_POST["repassword"]

    );
    $db->insertUser($u);
    $criptedMail = $_POST["email"] ^ $privateKey;
    $message = '<hr>How are you? <br> <hr>This is your link:<a href="http://cashyland.tk/php/login/validate.php?id=' . urldecode($criptedMail) . '">Click me!</a>';
    $subject = "Hi there! Verify your email :)";
    $mailSender->mailSend($_POST["email"], $subject, $message);
    header("Location: ../../verifyMail.html");
```

3.3 Test front-end (Selenium + JUnit)

L'installazione di Selenium nel server è fondamentale per far andare i test nel server.

Per fare ciò andiamo ad utilizzare Maven, che scaricherà tutti i collegamenti Java e tutte le sue dipendenze utilizzando il file maven pom.

Il primo passo del processo per l'installazione di Selenium è quindi, quello di creare una cartella che conterrà i file del progetto Selenium, per poi creare il file pom.xml, che conterrà le configurazioni del progetto.

```
<? xml version="1.0" encoding="UTF-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns="http://maven.apache.org/POM/4.0.0"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>SeleniumTest</groupId>
    <artifactId>SeleniumTest</artifactId>
    <version>1.0</version>
    <dependencies>
        <dependency>
            <groupId>org.seleniumhq.selenium </groupId>
            <artifactId>selenium-server</artifactId>
            <version>3.0.1</version>
        </dependency>
        <dependency>
            <groupId>org.seleniumhq.selenium </groupId>
            <artifactId>htmlunit-driver</artifactId>
            <version>2.34.0</version>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.5.0-M1</version>
        </dependency>
    </dependencies>
```

```
</dependency>
</dependencies>
</project>
```

Dopo di ciò si usa il comando "`mvn clean install`" dalla riga di comando dentro la directory del progetto, questo scaricherà Selenium, tutte le sue dipendenze e le aggiungerà al progetto.

Il passo successivo è quello di importare il progetto maven in IntelJ e creare i test junit5(jupiter) che successivamente verranno spostati nel Server.

Ora nel server andremo a installare tutto quello che è necessario per utilizzare i test di Selenium.

Si utilizza il seguente comando per installare Xvfb (X virtual framebuffer), questo implementa il protocollo del display server X11 senza alcun display.

```
sudo apt-get install -y unzip xvfb libxi6 libgconf-2-4
```

Usiamo il seguente comando sottostante per installare OpenJDK.

```
sudo apt-get install default-jdk
```

A questo punto vengono installate tutte le librerie nel server di junit5(jupiter).

```
apiguardian-api-1.0.0.jar
junit-jupiter-api-5.4.2.jar
junit-platform-commons-1.4.2.jar
opentest4j-1.1.1.jar
geckodriver-v0.24.0-linux64
```

Insieme a Firefox e il suo driver Geckodriver (geckodriver-v0.24.0-linux64) coi seguenti comandi.

```
sudo apt-get install firefox
wget https://github.com/mozilla/geckodriver/releases/download/v0.24.0/geckodriver-v0.24.0-linux64.tar.gz
tar -xvf geckodriver*
chmod +x geckodriver
```

Il Selenium Server è necessario per l'esecuzione dei Web Driver Selenium da remoto, è necessario scaricare il file jar del server stand-alone Selenium (`selenium-server-standalone-3.13.0.jar`) utilizzando il seguente comando.

```
wget https://selenium-release.storage.googleapis.com/3.13/selenium-server-standalone-3.13.0.jar
```

Inoltre è necessario installare la libreria testng-6.8.7.jar, che verrà installata coi seguenti comandi.

```
wget http://www.java2s.com/Code/JarDownload/testng/testng-6.8.7.jar.zip
unzip testng-6.8.7.jar.zip
```

Successivamente sarà necessario avere le librerie nella stessa directory dei test java, dove prima che verranno eseguiti, si dovranno esportare le librerie col comando `export classpath`, che verrà utilizzato in questo modo.

```
export CLASSPATH=".
  : selenium-server-standalone-3.13.0.jar
  : testng-6.8.7.jar
  : apiguardian-api-1.0.0.jar
  : junit-jupiter-api-5.4.2.jar
  : junit-platform-commons-1.4.2.jar
  : opentest4j-1.1.1.jar"
```

La scrittura dei test di Selenium era in parte una novità perché tutti i membri del gruppo conoscevano già il linguaggio Java ma nessuno aveva mai utilizzato un WebDriver. A causa di questa mancanza è stata necessaria una parte di raccolta di informazioni che ha prolungato leggermente questa parte del progetto. I

test sono abbastanza intuitivi e facili da creare, bisogna innanzitutto creare una classe di che conterrà solamente il metodo main e che richiamerà i test.

```
public class SeleniumTest {  
    public static void main(String[] args) {  
        SeleniumTestTest st = new SeleniumTestTest();  
        st.test();  
    }  
}
```

Fatto ciò si potrà, premendo la combinazione di tasti **ctrl+shift+t**, creare una classe di test. Essa conterrà tutto il codice necessario a Selenium per controllare il corretto funzionamento delle pagine web. Il test verrà eseguito sulla base del framework JUnit5 dato che esso è il più utilizzato per il linguaggio scelto.

Come prima cosa all'interno della classe bisognerà dichiarare quale è l'URL del sito di cui si vogliono eseguire i test e il WebDriver che per il momento resterà di valore nullo.

```
String URL = "http://cashyland.tk/";  
WebDriver driver = null;
```

Dopodiché si dovrà nel metodo test specificare la posizione del WebDriver ed in seguito istanziare lo stesso.

```
final File firefoxPath = new File(System.getProperty("lmportal.deploy.firefox.path",  
"/usr/bin/firefox"));  
driver = new FirefoxDriver();
```

Una volta istanziato il driver si dovrà bloccare il codice finché la pagina viene caricata completamente altrimenti non si potrà interagire con gli elementi del sito.

```
WebDriverWait wait = new WebDriverWait(driver, 60);  
wait.until(ExpectedConditions.elementToBeClickable(By.className("container"))));
```

Fatto ciò si potrà usare qualsiasi elemento della pagina semplicemente creando un WebElement ed assegnandogli l'elemento corrispondente tramite uno degli svariati metodi che consentono di trovare un tag. Essi consentono di trovare un componente della pagina tramite uno dei suoi attributi, in base al testo che contiene, alle sue dimensioni e così via per innumerevoli possibilità. Una volta trovato il tag cercato potremo eseguire molte operazioni con esso come premerlo, eseguirne l'hover, scriverci del testo e via dicendo.

```
WebElement home = driver.findElement(By.linkText("Home"));  
home.click();  
assertEquals("CashyLand - Home", driver.getTitle());
```

Con il WebElement potremo inoltre prendere le informazioni dell'elemento come il titolo, la dimensione, le coordinate e molto altro per controllare che siano quelle che desideriamo con degli assert.

Alla fine del programma bisognerà chiudere il driver con il comando:

```
driver.quit();
```

I test possono essere eseguiti sia con l'interfaccia grafica che mostra nel browser cosa sta facendo il test oppure senza di essa se ad esempio si sta utilizzando un sistema operativo senza GUI.

Per far funzionare i test con l'interfaccia basterà farli partire dal IDE che si sta utilizzando.

Per eseguire i test senza interfaccia grafica bisognerà utilizzare il driver web di Xvfb che è di base un sistema headless. Utilizzando esso basterà poi seguire questa procedura:

Innanzitutto andare nella cartella dove sono contenuti i tests e fare partire il server con il comando:

```
Xvfb :0 &
```

Fatto ciò bisognerà salvare il numero dello schermo in una variabile di sistema, questo verrà fatto con il seguente comando:

```
export DISPLAY=:0
```

Dopodiché si potrà eseguire il file che contiene il metodo main e i test dovrebbero funzionare, se questo non è vero bisognerà fermare i processi già in esecuzione di firefox, selenium e del geckodriver per poi riprovare la procedura. Per bloccare i processi si possono usare i seguenti tre comandi:

```
pkill selenium  
pkill geckodriver  
pkill firefox
```

3.4 Test automatizzati con Jenkins

Per eseguire i test in maniera automatica si utilizzerà Jenkins. Per la guida all'installazione e configurazione del software vedi allegati

"2019.02.22_i3_davidi_dueblin_forni_pezzotti_toscanelli_installazione_configurazione_jenkins".

3.5 Test back-end (PHPUnit)

3.6 Database (MySQL)

Il database che contiene tutti i dati utilizzati dall'applicazione per offrire ai suoi utenti un'esperienza su misura è stato realizzato con MySQL e segue alla lettera la progettazione effettuata (vedi Design dei dati e database). Il codice utilizzato per la realizzazione è contenuto nel file [/code/sql/DB/cashyland_db_2.sql](#), mentre gli identificatori esterni (foreign keys) sono stati inseriti all'interno del file [/code/sql/DB/foreign_keys.sql](#). Di seguito tutti i codici utilizzati per la creazione delle strutture delle tabelle. L'ordine di descrizione delle tabelle è quello in cui è scritto il codice ed è necessario tenerlo tale per evitare conflitti con le foreign keys alla creazione di una tabella.

3.6.0 Creazione database

Il database in sé, contenente tutte le tabelle, è stato creato utilizzando il seguente codice:

```
DROP DATABASE IF EXISTS `cashyland`;  
CREATE DATABASE `cashyland`;
```

Mentre viene poi impostato come database da usare con il comando successivo:

```
USE `cashyland`;
```

3.6.1 Room

La tabella 'room' è quella che contiene le informazioni relative alle varie sale all'interno del database, ovvero 'location', che ne rappresenta il nome e 'description' che ne contiene una descrizione. È stata creata con questo codice:

```
DROP TABLE IF EXISTS `room`;  
  
CREATE TABLE `room` (  
    `location` varchar(45) NOT NULL,  
    `description` varchar(256) DEFAULT NULL,  
    PRIMARY KEY (`location`)  
) ;
```

3.6.2 Game

La tabella 'game' contiene i dati relativi ai vari giochi in cui è possibile dilettersi all'interno del casinò cashyland. La colonna 'room' deve contenere un valore contenuto all'interno della tabella 'room', sotto la colonna 'location'. Il codice utilizzato per creare la tabella è il seguente:

```
DROP TABLE IF EXISTS `game`;

CREATE TABLE `game` (
  `room` varchar(45) NOT NULL,
  `name` varchar(45) NOT NULL,
  `description` varchar(256) DEFAULT NULL,
  PRIMARY KEY (`name`, `room`),
  KEY `fk_room_idx` (`room`),
  CONSTRAINT `fk_room` FOREIGN KEY (`room`) REFERENCES `room` (`location`) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

3.6.3 Media type

All'interno della tabella 'media_type' si trovano i valori predefiniti da poter assegnare a un'immagine, per esempio "Immagine" o "Video". Il codice che crea la tabella è il seguente:

```
DROP TABLE IF EXISTS `media_type`;

CREATE TABLE `media_type` (
  `name` varchar(45) NOT NULL,
  PRIMARY KEY (`name`)
);
```

3.6.4 Media

La tabella 'media' contiene i link riferiti ai vari file multimediali che devono essere mostrati all'interno del sito web e una colonna con un identificatore esterno che fa riferimento alla tabella 'media_type' (vedi Media type). Il codice che crea la tabella è il successivo:

```
DROP TABLE IF EXISTS `media`;

CREATE TABLE `media` (
  `url` varchar(100) NOT NULL,
  `type` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`url`),
  KEY `type` (`type`),
  CONSTRAINT `media_ibfk_1` FOREIGN KEY (`type`) REFERENCES `media_type` (`name`) ON DELETE SET NULL
ON UPDATE CASCADE
);
```

3.6.5 Game media

La tabella 'game_media' è quella al cui interno si trovano i collegamenti tra un gioco e un file multimediale. Infatti, questa tabella-ponte è composta da due identificatori esterni, uno che punta a 'game'.'name' (vedi Game) e l'altro che fa riferimento a 'media'.'url' (vedi Media). Viene creata con questo codice:

```
DROP TABLE IF EXISTS `game_media`;
```

```
CREATE TABLE `game_media` (
  `game_name` varchar(45) NOT NULL,
  `media_url` varchar(100) NOT NULL,
  PRIMARY KEY (`game_name`, `media_url`),
  KEY `game_media_media_idx` (`media_url`),
  CONSTRAINT `game_media_ibfk_1` FOREIGN KEY (`media_url`) REFERENCES `media` (`url`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `game_media_ibfk_2` FOREIGN KEY (`game_name`) REFERENCES `game` (`name`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

3.6.6 Gender

La tabella 'gender' contiene i valori predefiniti per il sesso di un utente che si registra alla piattaforma, ossia "Maschio" e "Femmina". Viene creata tramite il codice seguente:

```
DROP TABLE IF EXISTS `gender`;
```

```
CREATE TABLE `gender` (
  `name` varchar(10) NOT NULL,
  PRIMARY KEY (`name`)
);
```

3.6.7 Promotion

All'interno della tabella 'promotion' si trovano i dati relativi alle promozioni mostrate ai vari tipi di utenti, ovvero un nome e una descrizione. Quello successivo è il codice utilizzato per generare la struttura della tabella:

```
DROP TABLE IF EXISTS `promotion`;
```

```
CREATE TABLE `promotion` (
  `id` int(11) NOT NULL,
  `name` varchar(45) DEFAULT NULL,
  `description` varchar(256) DEFAULT NULL,
  PRIMARY KEY (`id`)
);
```

3.6.8 Promotion media

La tabella 'promotion_media' è un'altra tabella-ponte tra le tabelle 'promotion' e 'media'. Infatti è composta da identificatori esterni, uno che si riferisce a 'promotion'.'id' (vedi Promotion) e l'altro che punta a 'media'.'url' (vedi Media). Di seguito il codice che crea la struttura della tabella:

```
DROP TABLE IF EXISTS `promotion_media`;
```

```
CREATE TABLE `promotion_media` (
  `promotion_id` int(11) NOT NULL,
  `media_url` varchar(100) NOT NULL,
  PRIMARY KEY (`promotion_id`, `media_url`),
  KEY `media_url` (`media_url`),
  CONSTRAINT `promotion_media_ibfk_1` FOREIGN KEY (`promotion_id`) REFERENCES `promotion` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `promotion_media_ibfk_2` FOREIGN KEY (`media_url`) REFERENCES `media` (`url`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

3.6.9 User type

La tabella ‘user_type’ contiene i valori predefiniti per i vari tipi di utente, ai quali vengono mostrate o meno le promozioni, per esempio “occasionale” o “settimanale”. Il codice che la genera è il seguente:

```
DROP TABLE IF EXISTS `user_type`;
```

```
CREATE TABLE `user_type` (
  `name` varchar(45) NOT NULL,
  PRIMARY KEY (`name`)
);
```

3.6.10 User

La tabella ‘user’ contiene i dati relativi a un utente, come il nome, il cognome, la password, la data di nascita, il sesso, che fa riferimento alla tabella ‘gender’, il tipo, che si riferisce alla tabella ‘user_type’, ... Segue il codice utilizzato per la sua creazione.

```
DROP TABLE IF EXISTS `user`;
```

```
CREATE TABLE `user` (
  `name` varchar(45) DEFAULT NULL,
  `surname` varchar(45) DEFAULT NULL,
  `street` varchar(45) DEFAULT NULL,
  `house_number` varchar(4) DEFAULT NULL,
  `zip_code` varchar(5) DEFAULT NULL,
  `city` varchar(45) DEFAULT NULL,
  `email` varchar(45) NOT NULL,
  `phone_number` varchar(15) DEFAULT NULL,
  `gender` varchar(10) DEFAULT NULL,
  `password` varchar(45) DEFAULT NULL,
  `type` varchar(45) DEFAULT NULL,
  `admin` tinyint(1) DEFAULT 0,
  `birthday` date DEFAULT NULL,
  `verified` tinyint(1) DEFAULT 0,
  PRIMARY KEY (`email`),
  KEY `gender` (`gender`),
  KEY `type` (`type`),
  CONSTRAINT `user_ibfk_1` FOREIGN KEY (`gender`) REFERENCES `gender` (`name`) ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT `user_ibfk_2` FOREIGN KEY (`type`) REFERENCES `user_type` (`name`) ON DELETE SET NULL ON UPDATE CASCADE
);
```

3.6.11 Promotion user

La tabella-ponte ‘promotion_user’ collega un certo tipo di utenti, che deve essere contenuto in almeno una riga di ‘user’.’type’, ad una promozione, contenuta all’interno di ‘promotion’.’id’. Viene creata con il seguente codice:

```
DROP TABLE IF EXISTS `promotion_user`;
```

```
CREATE TABLE `promotion_user` (
  `user_type` varchar(45) NOT NULL,
  `promotion_id` int(11) NOT NULL,
  PRIMARY KEY (`user_type`,`promotion_id`),
  KEY `promotion_id` (`promotion_id`),
  CONSTRAINT `promotion_user_ibfk_1` FOREIGN KEY (`user_type`) REFERENCES `user` (`type`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `promotion_user_ibfk_2` FOREIGN KEY (`promotion_id`) REFERENCES `promotion` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

3.6.12 Room media

La tabella 'room_media' è una tabella-ponte che collega una stanza, rappresentata da un identificatore esterno che fa riferimento a 'room'. 'location' a un file multimediale il cui riferimento è contenuto all'interno della tabella 'media'. 'url'. Questa tabella viene generata tramite il codice successivo:

```
DROP TABLE IF EXISTS `room_media`;
```

```
CREATE TABLE `room_media` (
  `room_location` varchar(45) NOT NULL,
  `media_url` varchar(100) NOT NULL,
  PRIMARY KEY (`room_location`, `media_url`),
  KEY `media_url` (`media_url`),
  CONSTRAINT `room_media_ibfk_1` FOREIGN KEY (`room_location`) REFERENCES `room` (`location`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `room_media_ibfk_2` FOREIGN KEY (`media_url`) REFERENCES `media` (`url`) ON DELETE
CASCADE ON UPDATE CASCADE
);
```

4 Test

4.1 Protocollo di test

Test Case:	TC-001	Nome:	Test di Navigazione		
Riferimento:	REQ-001				
Descrizione:	Questo test naviga attraverso tutte le pagine web di cashyland.tk, verificando che i vari link che portano alle altre sezioni del sito funzionino correttamente.				
Prerequisiti:	Selenium e Java installati con tutte le librerie necessarie per i test (selenium-server-standalone-3.13.0.jar, testng-6.8.7.jar, apiguardian-api-1.0.0.jar, junit-jupiter-api-5.4.2.jar, junit-platform-commons-1.4.2.jar, opentest4j-1.1.1.jar).				
Procedura:	<p>Dalla Pagina principale del sito (Home) seguire i passi descritti di seguito:</p> <ol style="list-style-type: none"> Cliccare sul link “Accedi”, che porta alla pagina di Login. Premere il pulsante “REGISTRATI”, che porta alla pagina di Registrazione. Selezionare il link “Accedi”, che porta alla pagina di Login. Seguire il link “Hai dimenticato la password?”, che porta alla pagina della password smarrita. Premere il link “Accedi”, che porta alla pagina di Login. Selezionare il link “Home”, che porta alla pagina principale. Cliccare sul link “Giochi”, che porta alla pagina dei giochi. Premere il link “Home”, che porta alla pagina principale. Seguire il link “Sale”, che porta alla pagina delle sale. Selezionare il link “Home”, che porta alla pagina principale. 				
Risultati attesi:	Non si presenta alcun errore durante la navigazione e, dopo aver eseguito la procedura dall'inizio alla fine, ci si ritrova sulla pagina principale del sito.				

Gestione casinò

Test Case:	TC-002	Nome:	Test di Registrazione
Riferimento:	REQ-005		
Descrizione:	Questo test naviga fino alla pagina di registrazione e verifica i vincoli di ogni input, per poi inserire delle credenziali valide.		
Prerequisiti:	Selenium e Java installati con tutte le librerie necessarie per i test (selenium-server-standalone-3.13.0.jar, testng-6.8.7.jar, apiguardian-api-1.0.0.jar, junit-jupiter-api-5.4.2.jar, junit-platform-commons-1.4.2.jar, opentest4j-1.1.1.jar).		
Procedura:	<p>Dalla pagina principale del sito, seguire i passi descritti in seguito:</p> <ol style="list-style-type: none"> 1. Scegliere il link “Accedi”, che porta alla pagina di accesso. 2. Inserire il valore non valido “!” nel campo “Nome” e viene premere il pulsante “Registrati”. 3. Inserire il valore non valido “123456789012345678901234567890123456789012345678901” nel campo “Nome” e viene premere il pulsante “Registrati”. 4. Ripetere i passi 2 e 3 anche per il campo “Cognome” 5. Vengono inseriti valori non validi nel campo “Data Nascita” e viene cliccato il pulsante “Registrati”. 6. Vengono inseriti valori non validi nel campo “Via” e viene cliccato il pulsante “Registrati”. 7. Vengono inseriti valori non validi nel campo “No. Civico” e viene cliccato il pulsante “Registrati”. 8. Vengono inseriti valori non validi nel campo “CAP” e viene cliccato il pulsante “Registrati”. 9. Vengono inseriti valori non validi nel campo “Città” e viene cliccato il pulsante “Registrati”. 10. Vengono inseriti valori non validi nel campo “No. Telefono” e viene cliccato il pulsante “Registrati”. 11. Vengono inseriti valori non validi nel campo “Email” e viene cliccato il pulsante “Registrati”. 12. Vengono inseriti valori non validi nel campo “Password”, “Ripeti Password” viene cliccato il pulsante “Registrati”. 13. Vengono inserite credenziali valide in tutti i campi e viene cliccato il pulsante “Registrati” 		
Risultati attesi:	Non si presenta alcun errore durante la navigazione, nella fase di test dei vincoli non si riesce a passare alla pagina di verifica mail, dopo aver inserito le credenziali valide si accede alla pagina di verifica mail.		

Test Case:	TC-003	Nome:	Test di Login
Riferimento:	REQ-006		
Descrizione:	Questo test naviga fino alla pagina di login, effettua la registrazione con un account già registrato e testa le varie funzionalità dell'utente.		
Prerequisiti:	Selenium e Java installati con tutte le librerie necessarie per i test (selenium-server-standalone-3.13.0.jar, testng-6.8.7.jar, apiguardian-api-1.0.0.jar, junit-jupiter-api-5.4.2.jar, junit-platform-commons-1.4.2.jar, opentest4j-1.1.1.jar).		
Procedura:	<p>Dalla pagina principale del sito, seguire i passi descritti in seguito:</p> <ol style="list-style-type: none"> 1. Scegliere il link “Accedi”, che porta alla pagina di accesso. 2. Inserire i valori “admin” nel campo email e “Password&1” nel campo password e cliccare il pulsante “Login”. Successivamente verrà verificato il funzionamento delle funzionalità dell’utente amministratore che ha effettuato l’accesso (l’utente dovrebbe avere i permessi per gestire utenti, sale, giochi, promozioni e immagini): 3. Premere il pulsante “Modifica Utenti”, per poi tornare alla pagina di gestione. 4. Viene cliccato il pulsante “Modifica Sale”, per poi tornare alla pagina di gestione. 5. Viene cliccato il pulsante “Modifica Giochi”, per poi tornare alla pagina di gestione. 6. Viene cliccato il pulsante “Modifica Promozioni”, per poi tornare alla pagina di gestione. 7. Viene cliccato il pulsante “Aggiungi File”. 		
Risultati attesi:	Non si presenta alcun errore durante la navigazione, si riesce a effettuare l’accesso, si riesce ad accedere alle pagine di gestione del sito (Gestione Utenti, Sale, Giochi, Promozioni e Aggiungi Immagine).		

Gestione casinò

Test Case:	TC-004	Nome:	Test classe User
Riferimento:	REQ-005		
Descrizione:	Questo va a verificare che la classe user sia corretta..		
Prerequisiti:	PHP e PHPUnit installati		
Procedura:	<p>Verificare che la registrazione di utente sia corretta.</p> <ol style="list-style-type: none"> 1. Provare a richiamare il metodo statico per la prova dell'email passandoli una mail valida 2. Provare a richiamare il metodo statico per la prova dell'email passandogli una mail non valida aspettandosi che quest'ultima tiri un exception 3. Provare a richiamare il metodo statico per la prova del nome passandoli un nome valido 4. Provare a richiamare il metodo statico per la prova del nome passandogli un nome non valida aspettandosi che quest'ultimo tiri un exception 5. Provare a richiamare il metodo che definisce l'età in base a una data passata. 6. Provare a richiamare il metodo statico per la prova della data passandoli una data valida 7. Provare a richiamare il metodo statico per la prova della data passandogli una data non valida aspettandosi che quest'ultima tiri un exception 8. Provare a richiamare il metodo statico per la prova del numero di casa passandogli un numero di casa valido 9. Provare a richiamare il metodo statico per la prova del numero di casa passandogli un numero di casa non valido aspettandosi che tiri un exception 10. Provare a richiamare il metodo statico per la prova del numero di telefono passandogli un telefono di casa valido 11. Provare a richiamare il metodo statico per la prova del numero di telefono passandogli un numero di telefono non valido aspettandosi che tiri un exception 12. Provare a richiamare il metodo statico per la prova del sesso passandogli un sesso valido 13. Provare a richiamare il metodo statico per la prova del sesso passandogli un sesso non valido aspettandosi che tiri un exception 14. Provare ad istanziare un nuovo user con dati corretti 15. Provare ad istanziare un nuovo con dati non corretti aspettandosi che quest'ultimo non venga istanziato. 		
Risultati attesi:	Non si presenta alcun problema legato a PHPUnit.		

Gestione casinò

Test Case:	TC-005	Nome:	Test classe SendMail
Riferimento:	REQ-005		
Descrizione:	Questo va a verificare che la class SendMail sia corretta. .		
Prerequisiti:	PHP e PHPUnit installati		
Procedura:	Verificare che l'invio delle mail sia corretto. 1. Provare a inviare una mail mailSend("[mail]","[subject]","[text]") se la mail venisse spedita allora il metodo ritorna true.		
Risultati attesi:	Non si presenta alcun problema legato a PHPUnit.		

Test Case:	TC-006	Nome:	Test classe Database
Riferimento:			
Descrizione:	Questo va a verificare che la class Database sia corretta. .		
Prerequisiti:	PHP e PHPUnit installati La classe user funzionante TC-00x		
Procedura:	Verificare che la connessione e l'esecuzione delle query sia corretta 1. Provare ad istanziare un oggetto database passandogli i dati legati al database 2. Provare ad eseguire una query con il metodo fornito dalla classe		
Risultati attesi:	Non si presenta alcun problema legato a PHPUnit.		

4.2 Risultati test

Test Case	Risultato
TC-001	Dopo essere arrivato alla home della pagina principale il test si blocca.
TC-002	Dopo essere arrivato alla home della pagina principale il test si blocca.
TC-003	Dopo essere arrivato alla home della pagina principale il test si blocca.
TC-004	Funziona correttamente
TC-005	Funziona correttamente
TC-006	Funziona correttamente

Gestione casinò

4.3 Mancanze/limitazioni conosciute

All'interno della pagina principale che si apre appena si visita il sito c'è un pulsante che legge "Scopri di più" che porta più in basso nella pagina, però per gli utenti che la visitano utilizzando Google Chrome questa funzionalità non funziona.

5 Consuntivo

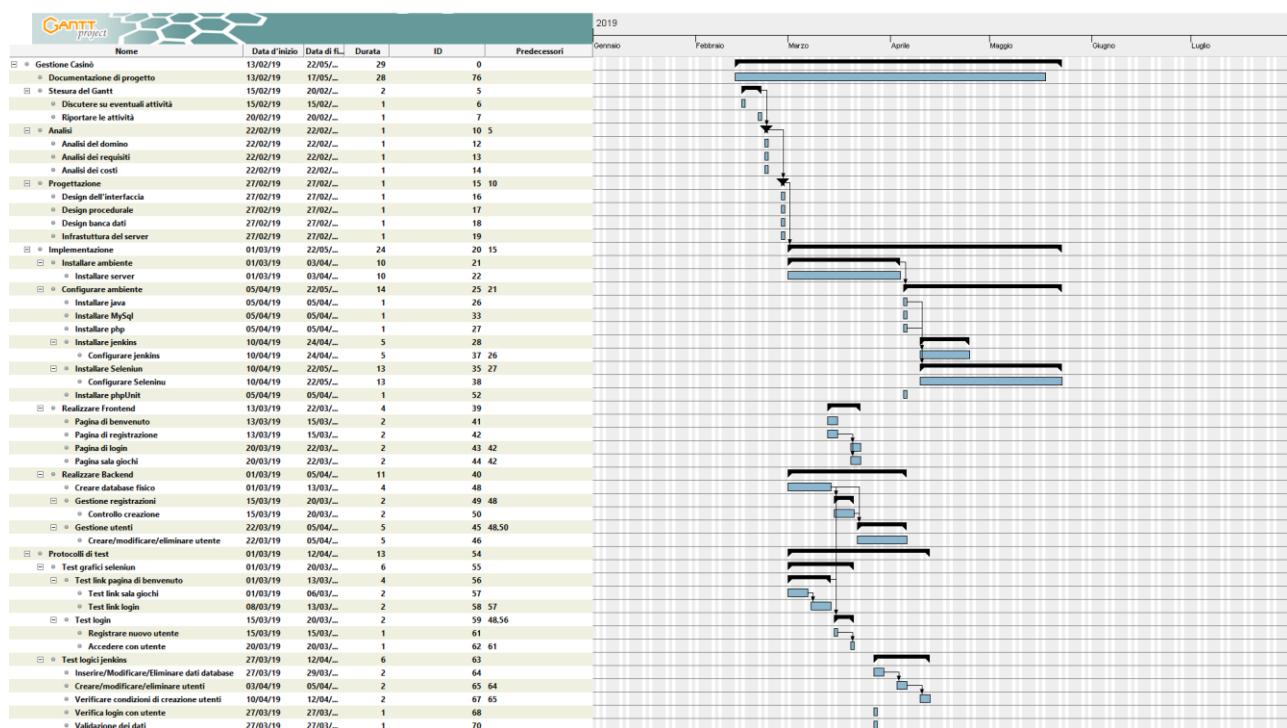


Figura 31 - Diagramma di Gantt consuntivo

6 Conclusioni

Grazie al nostro prodotto il casinò CashyLand ha la possibilità di farsi pubblicità online, una cosa indispensabile soprattutto in tempi recenti, il che attrarrà potenziali clienti e porterà a nuovi potenziali guadagni. Senza contare che il lato della gestione, sempre attraverso il sito web da noi realizzato, che permette di aggiungere, rimuovere e modificare facilmente giochi, sale e promozioni, in modo che non bisogna essere a conoscenza di linguaggi di scripting o di interrogazione database, ma basta accedere alla piattaforma con un account con diritti di amministratore.

6.1 Sviluppi futuri

Aggiungere i test di Selenium al server di automazione di Jenkins per verificare il corretto funzionamento dell'interfaccia grafica e delle pagine web attraverso tutta l'applicazione dopo ogni push sul repository di GitHub in modo da rendere l'integrazione continua più completa.

6.2 Considerazioni personali

Grazie a questo progetto abbiamo imparato almeno le basi dell'utilizzo di software che vengono usati anche nel mondo del lavoro per l'integrazione continua e i test, come Selenium e Jenkins. Abbiamo quindi integrato un framework di test, come JUnit e PHPUnit, con un server di automazione come Jenkins in modo che la produzione di codice corretto e di qualità risulti più facile e, soprattutto, non rischi di danneggiare il funzionamento del codice già presente.

7 Bibliografia

7.1 Sitografia

- <https://medium.com/@khandelwalnidhi/jenkins-setup-for-php-unit-testing-on-aws-c39baad7a99e>, Jenkins Setup for PHPUnit, 13.02.2019
- <https://stackoverflow.com/questions/39621263/jenkins-fails-when-running-service-start-jenkins>, Jenkins fail to run, 13.02.2019
- <https://www.ubuntu.com/download/server>, Install ubuntu server, 13.02.2019
- <https://thishosting.rocks/install-php-on-ubuntu/>, Install PHP on ubuntu, 13.02.2019
- <https://linuxize.com/post/how-to-install-jenkins-on-ubuntu-18-04/>, Install jenkins on ubuntu, 13.02.2019
- <https://notifyjs.jpillora.com/>, Notify.js, 13.03.2019
- <https://tecadmin.net/setup-selenium-chromedriver-on-ubuntu/>, Install and setup Selenium, 15.02.2019
- https://www.tutorialspoint.com/selenium/selenium_webdriver.htm, Selenium Webdriver, 15.03.2019
- <https://www.seleniumhq.org/projects/webdriver/>, Selenium Webdriver, 15.03.2019
- <https://www.built.io/blog/run-selenium-tests-in-headless-browser>, Run Selenium Tests In Headless Browser - Built.io Blog, 15.03.2019
- <https://wiki.saucelabs.com/>, Getting Started with Selenium for Automated Website Testing - The Sauce Labs Cookbook - Sauce Labs Documentation Wiki, 15.03.2019
- https://www.tutorialspoint.com/groovy/groovy_unit_testing.htm, Groovy Unit Testing, 15.03.2019
- <https://maven.apache.org/install.html>, Maven - Installing Apache Maven, 20.03.2019
- <http://chromedriver.chromium.org/>, ChromeDriver - WebDriver for Chrome, 20.03.2019
- <https://github.com/PHPMailer/PHPMailer>, PHPMailer/PHPMailer: The classic email sending library for PHP, 22.03.2019
- <https://www.pexels.com/search/casino/>, 30+ Interesting Casino Photos · Pexels · Free Stock Photos, 22.03.2019
- <https://junit.org/junit5/>, JUnit 5, 29.03.2019
- <https://mediatemple.net/community/products/dv/204403864/export-and-import-mysql-databases>, Export and import MySQL databases - Media Temple, 12.04.2019
- <https://www.guru99.com/accessing-forms-in-webdriver.html>, Selenium Form WebElement: TextBox, Submit Button, sendkeys(), click(), 12.04.2019
- <https://serverfault.com/questions/548996/syntax-error-unknown-user-munin-in-statoverride-file>, dpkg - syntax error: unknown user 'munin' in statoverride file - Server Fault, 17.04.2019
- <https://tecadmin.net/setup-selenium-chromedriver-on-ubuntu/>, How to Setup Selenium with ChromeDriver on Ubuntu 18.04 & 16.04 – TecAdmin, 17.04.2019
- <https://gist.github.com/ziadoz/3e8ab7e944d02fe872c3454d17af31a5>, Install Chrome, ChromeDriver and Selenium on Ubuntu 16.04 · GitHub, 17.04.2019
- <https://www.draw.io/>, draw.io, 10.05.2019

Allegati

Elenco degli allegati, esempio:

- Diari di lavoro
- Guida di installazione e configurazione Jenkins

Installazione e configurazione di Jenkins

Sommario

Installazione Jenkins.....	3
Configurazione Jenkins	3
Creare un lavoro/elemento su Jenkins	4
Configurare GitHub per l'utilizzo con Jenkins.....	6
Indice delle figure.....	8

Installazione Jenkins

Per installare Jenkins su Linux ubuntu server 18.04 sarà necessario seguire la seguente procedura.

Si inizierà aggiungendo la chiave del repository di Jenkins con il seguente comando:

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
```

Dopodiché si aggiungerà il repository alle liste dei sorgenti del server con questo comando:

```
echo deb https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list
```

Poi bisognerà aggiornare il server così che utilizzi la nuova cartella:

```
sudo apt-get update
```

Infine potremo installare Jenkins digitando semplicemente il seguente comando:

```
sudo apt-get install jenkins
```

Per attivare il servizio in caso questo non sia stato fatto automaticamente si dovrà utilizzare il seguente comando:

```
sudo systemctl start jenkins
```

Configurazione Jenkins

Il primo passaggio per configurare Jenkins sarà quello di collegarsi alla dashboard di quest'ultimo, per fare ciò si digiterà su un browser l'ip del server seguito dalla porta di ascolto di Jenkins (solitamente la 8080) come nell'esempio seguente:

<ip_server>:8080

Al primo accesso verrà richiesta una password per sbloccare Jenkins e come riportato nella finestra essa la potremo trovare al seguente percorso, per aprire il file si utilizzerà il comando cat:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



Figura 1 Password di sblocco per Jenkins

Fatto ciò vedremo due tasti che ci daranno uno la possibilità di installare i plugin consigliati mentre uno di scegliere noi quali installare, selezionando il primo verrà automaticamente installata la plugin di GitHub che interessa a noi quindi possiamo premere il primo bottone.

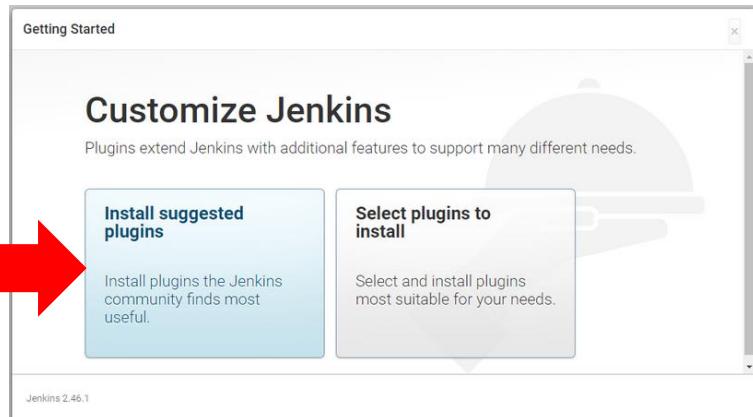


Figura 2 Installazione iniziale delle plugins di Jenkins

Dopodiché si dovrà creare un utente amministratore e finalmente Jenkins sarà pronto all'uso.

Creare un lavoro/elemento su Jenkins

Per creare un lavoro basterà, dalla pagina principale, premere su **Nuovo elemento**



Figura 3 Creazione di un nuovo elemento

Fatto ciò si dovrà selezionare **Progetto freestyle** e dargli un nome



Figura 4 Dare un nome e scegliere il tipo di elemento

Fatto ciò si aprirà una finestra di impostazioni dell'elemento.

Inizialmente vi sarà lo spazio per una descrizione e subito dopo vi sarà una serie di checkbox riguardanti l'impostazione dell'elemento, si selezionerà **GitHub project** e nel campo di testo che apparirà si dovrà inserire l'URL della repository di GitHub.



Elimina compilazioni precedenti
 GitHub project
Project url: [empty field]
 Questo progetto è parametrizzato
 This build requires lockable resources
 Throttle builds
 Disabilita questo progetto
 Esegui compilazioni parallele se necessario (beta)
 Avanzate...
 Avanzate...

Figura 5 Impostazione dell'elemento come progetto di GitHub

Fatto ciò nella parte seguente si sceglierà come gestione del codice sorgente **Git** e si inserirà nel campo di testo dell'**URL di deposito** il percorso della repository che si utilizza ad esempio per clonare la repository. Tutto il resto rimarrà invariato.



Gestione codice sorgente
 Nessuno
 Git
Depositi
URL di Deposito: [empty field] **Please enter Git repository.**
Credenziali: - none -
 Avanzate...

Rami a costruire
Ramo (lasciare in bianco per alcune): */master

Browser repository: (Automatico)
Comportamenti supplementari:
 Subversion
 Avanzate...

Figura 6 Gestione codice sorgente dell'elemento

Nella sezione successiva si andranno ad impostare i **Trigger di compilazione** cioè quando verranno eseguiti i test di Jenkins



Trigger compilazione
 Attiva una compilazione da remoto (ad esempio da uno script)
 Compila dopo aver compilato gli altri progetti
 Compila periodicamente
 GitHub Branches
 GitHub Pull Requests
 GitHub hook trigger for GITScm polling
 Esegui polling sul sistema di controllo del codice sorgente
 Avanzate...

Figura 7 Impostazione dei trigger di compilazione

La parte riguardante l'ambiente di compilazione potrà essere lasciata vuota e si andrà quindi a configurare il passaggio successivo riguardante la **Compilazione**, qui si selezionerà cosa dovrà fare Jenkins quando verrà attivato dal trigger impostato in precedenza. In questo caso si sceglierà **Esegui shell** e dentro di essa si metteranno i test phpUnit da eseguire.



Figura 8 Comandi da eseguire durante la compilazione

Fatto ciò la configurazione sarà terminata e potremo salvare.

Configurare GitHub per l'utilizzo con Jenkins

Per collegare Jenkins alla repository di GitHub bisognerà andare nelle impostazioni di quest'ultima e creare un webhook. Per fare ciò si dovrà appunto accedere al pannello di impostazioni della cartella di GitHub.

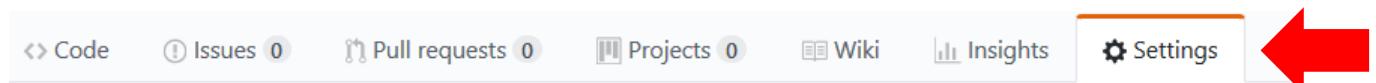


Figura 9 GitHub settings

Nella finestra che si aprirà si dovrà scegliere il menu riguardante i webhooks.

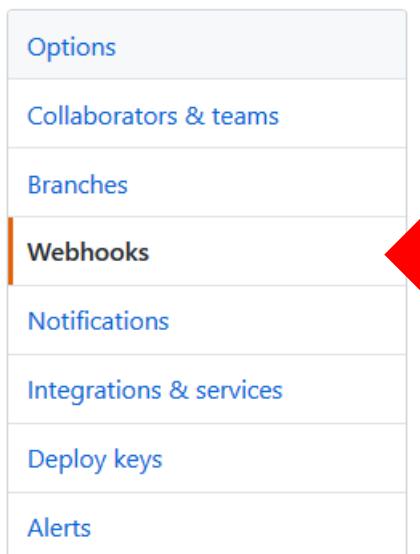


Figura 10 GitHub webhooks menu

Dopodiché, per creare un nuovo webhook, si selezionerà il tasto situato in alto a destra.



Figura 11 Bottone creazione nuovo webhook

Dopo aver premuto il bottone apparirà una finestra che andrà compilata come segue. Sotto **payload** si inserisce l'indirizzo del server seguito da "/github-webhook".

Payload URL *

http://<server_ip>/github-webhook/

Figura 12 Inserimento dell'URL del payload

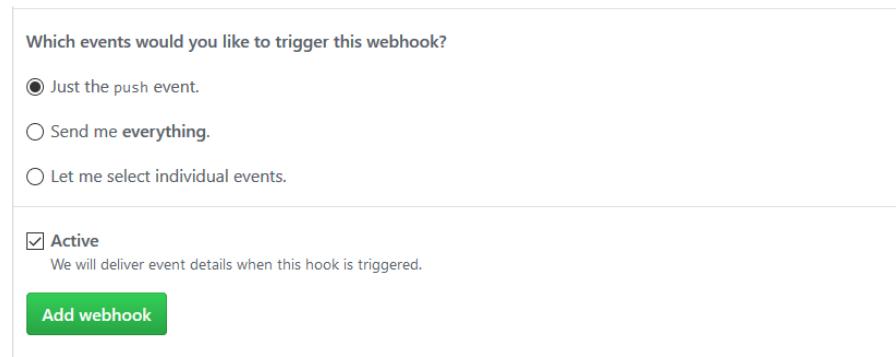
Nel campo **Content type** inserire "**application/json**".

Content type

application/json

Figura 13 Selezione del tipo del contenuto

Il campo **secret** potrà essere lasciato vuoto mentre sotto **Which events would you like to trigger this webhook?** selezionare il radio button con scritto "**Just the push event**".



A screenshot of a web-based form for creating a new webhook. The form includes fields for the payload URL, content type (set to application/json), and event triggers. The "Just the push event" option is selected. An "Active" checkbox is checked, with a descriptive note below it. A green "Add webhook" button is at the bottom.

Which events would you like to trigger this webhook?

Just the push event.
 Send me everything.
 Let me select individual events.

Active
We will deliver event details when this hook is triggered.

Add webhook

Figura 14 Webhook event trigger

Fatto ciò il collegamento con Jenkins dovrebbe funzionare, per testarne il corretto funzionamento si eseguirà un push alla cartella di GitHub ed esso dovrebbe far partire il lavoro di jenkins.

Indice delle figure

Figura 1 Password di sblocco per Jenkins	3
Figura 2 Installazione iniziale delle plugins di Jenkins	4
Figura 3 Creazione di un nuovo elemento.....	4
Figura 4 Dare un nome e scegliere il tipo di elemento	4
Figura 5 Impostazione dell'elemento come progetto di GitHub	5
Figura 6 Gestione codice sorgente dell'elemento	5
Figura 7 Impostazione dei trigger di compilazione	5
Figura 8 Comandi da eseguire durante la compilazione	6
Figura 9 GitHub settings	6
Figura 10 GitHub webhooks menu	6
Figura 11 Bottone creazione nuovo webhook	7
Figura 12 Inserimento dell'URL del payload	7
Figura 13 Selezione del tipo del contenuto	7
Figura 14 Webhook event trigger	7

GESTIONE CASINÒ | Diario di lavoro - 13.02.2019

Matan Davidi, Thor Düblin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 13 Febbraio 2019

Lavori svolti

Durante la giornata di oggi, la prima del nuovo progetto, il gruppo ha cominciato a informarsi sui software che avrebbe dovuto usare, ossia Jenkins e Selenium. Inoltre ognuno ha dovuto installare una macchina virtuale sul proprio PC con sistema operativo Ubuntu Server in modo da averlo pronto per i test. Mattia Toscanelli e Matteo Forni si sono occupati di informarsi e provare il funzionamento di Jenkins su Ubuntu server. Dopo qualche problema sono riusciti ad avviare il servizio Jenkins funzionante.

Carlo ha colmato le sue lacune riguardanti PHPUnit e ha realizzato un codice di prova per testare PHPUnit.

Nel gruppo pensavamo di suddividerci i compiti in questo modo:

- Matan -> progettazione del database e progettazione
- Thor -> Front-End e ricerca di informazioni su Selenium.
- Carlo -> Back-End con PHP (Login, connessione al DB e test)
- Matteo -> Ubuntu server con installazione e configurazione di Jenkins, Selenium e tutti i servizi necessari
- Mattia -> Ubuntu server con installazione e configurazione di Jenkins, Selenium e tutti i servizi necessari

Problemi riscontrati e soluzioni adottate

Il problema riscontrato con il funzionamento di Jenkins è che esso non supporta le versioni di java più recenti della 8 e quindi installando la 10 non funzionava più.

Punto della situazione rispetto alla pianificazione

Non abbiamo ancora fatto la pianificazione.

Programma di massima per la prossima giornata di lavoro

Nella prossima giornata vorremmo porre eventuali domande al docente più fare la progettazione.

GESTIONE CASINÒ | Diario di lavoro - 15.02.2019

Matan Davidi, Thor Düblin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 15 Febbraio 2019

Lavori svolti

Durante la giornata di oggi, il gruppo ha continuato le attività iniziate nella scorsa lezione.

Da questa lezione in avanti, assieme al datore di lavoro si è deciso di proseguire con un sistema di progettazione a quattro fasi con l'ausilio dei post-it. All'inizio di tutte le giornate di lavoro si discuterà del lavoro svolto l'ultima giornata e del programma per quella attuale e dei problemi riscontrati. Inoltre si separa il lavoro, rappresentato da un post-it in:

- to do, lavori ancora da svolgere
- work in progress, lavori in atto
- test, lavori terminati e in fase di test
- end, lavori terminati e testati

Matan per le prime due ore di lezione ha dovuto presentare il progetto precedente ai clienti, quindi non ha potuto lavorare. Le due ore successive, però, ha cominciato a realizzare una prima bozza del diagramma E/R per la progettazione del database, in modo da poterla mostrare al cliente e richiedere un feedback.

Thor ha installato la IDE di Selenium e ha iniziato a leggere la documentazione per capire come approcciarsi a questo software di automazione di test, dopo varie prove ha cominciato a comprendere come viene utilizzata la IDE, riscontrando un paio di problemi sui semplici test.

Successivamente si è occupato di installare Selenium WebDriver con l'utilizzo del software Maven.

Mattia si è occupato di continuare con la configurazione di Jenkins cercando di collegarsi con GitHub. Purtroppo però l'operazione non ha avuto successo, infatti è riuscito a trovare un metodo per collegarsi da GitHub al server Ubuntu. Quindi ha deciso di mettere in pausa questo Task e iniziare a cercare un template adatto per il sito.

Carlo oggi si è occupato di gestire lo standUp del gruppo (post-it attaccati al muro). Inoltre ha finito il diagramma di Gantt. Infine ha iniziato a programmare la i test della classe di registrazione degli utenti. Nel caso un utente non soddisfi i requisiti richiesti viene sollevata un eccezione di tipo InvalidArgumentException.

Matteo si è occupato, insieme a Carlo, di gestire lo standUp del gruppo, inoltre si è informato su come integrare Jenkins, GitHub e Selenium con i vari protocolli di test. Fatto ciò ha creato un primo schema dell'interfaccia delle pagine web su cui poi Mattia ha iniziato a lavorare.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Nella prossima lezione vorremo terminare il diagramma ER, capire come funziona Selenium, finire i test sulla classe di registrazione degli utenti e continuare con il front-end della pagina. Inoltre vorremo informarci ancora sui sistemi da integrare.

GESTIONE CASINÒ | Diario di lavoro - 20.02.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 20 Febbraio 2019

Lavori svolti

La giornata di lavoro di oggi si è divisa in questo modo:

Matan ha terminato il diagramma E/R per la progettazione del database e l'ha mostrato al cliente chiedendo un feedback.

Thor ha capito come funziona Selenium IDE, riuscendo a fare dei test automatizzati completamente senza errori, navigando liberamente sul sito di hotel Trivago. Successivamente si ha continuato l'installazione per vedere come funziona il WebDriver di Selenium.

Mattia Toscanelli ha continuato con l'adattamento della pagina html al contesto. Più precisamente ha lavorato sulla pagina di login e sulla pagina di registrazione. Carlo Pezzotti si è occupato di sviluppare tutti i test per la creazione di un nuovo user. I test sono sia modulari che globali. E ha sviluppato una classe user per gli utenti.

Problemi riscontrati e soluzioni adottate

Digitando da linea di comando "mvn clean install", ho avuto il seguente errore.

```
Seleziona Amministratore: Prompt dei comandi
Microsoft Windows [Versione 10.0.17134.598]
(c) 2018 Microsoft Corporation. Tutti i diritti sono riservati.
C:\Windows\system32>cd C:\Users\Utente\Desktop\selenium_projects
C:\Users\Utente\Desktop\selenium_projects>mvn clean install
[INFO] Scanning for projects...
[ERROR] [ERROR] Some problems were encountered while processing the POMs:
[ERROR] [FATAL] Non-parseable POM C:\Users\Utente\Desktop\selenium_projects\pom.xml: Unrecognised tag: 'Dependency' (position: START_TAG seen ...<dependencies>\r\n                                     <Dependency>... @7:25)  @ line 7, column 25
[ERROR] [FATAL] The build could not read 1 project -> [Help 1]
[ERROR] [INFO]   The project C:\Users\Utente\Desktop\selenium_projects\pom.xml has 1 error
[ERROR] [INFO]     Non-parseable POM C:\Users\Utente\Desktop\selenium_projects\pom.xml: Unrecognised tag: 'Dependency' (position: START_TAG seen ...<dependencies>\r\n                                     <Dependency>... @7:25)  @ line 7, column 25 -> [Help 2]
[ERROR] [INFO]     To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] [INFO]     Re-run Maven using the -X switch to enable full debug logging.
[ERROR] [INFO]     For more information about the errors and possible solutions, please read the following articles:
[ERROR] [INFO]       [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/ProjectBuildingException
[ERROR] [INFO]       [Help 2] http://cwiki.apache.org/confluence/display/MAVEN/ModelParseException
C:\Users\Utente\Desktop\selenium_projects>
```

Punto della situazione rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Carlo pezzotti andrà ad interfacciarsi con il database creato da Matan Davidi per inserire gli utenti nel database.

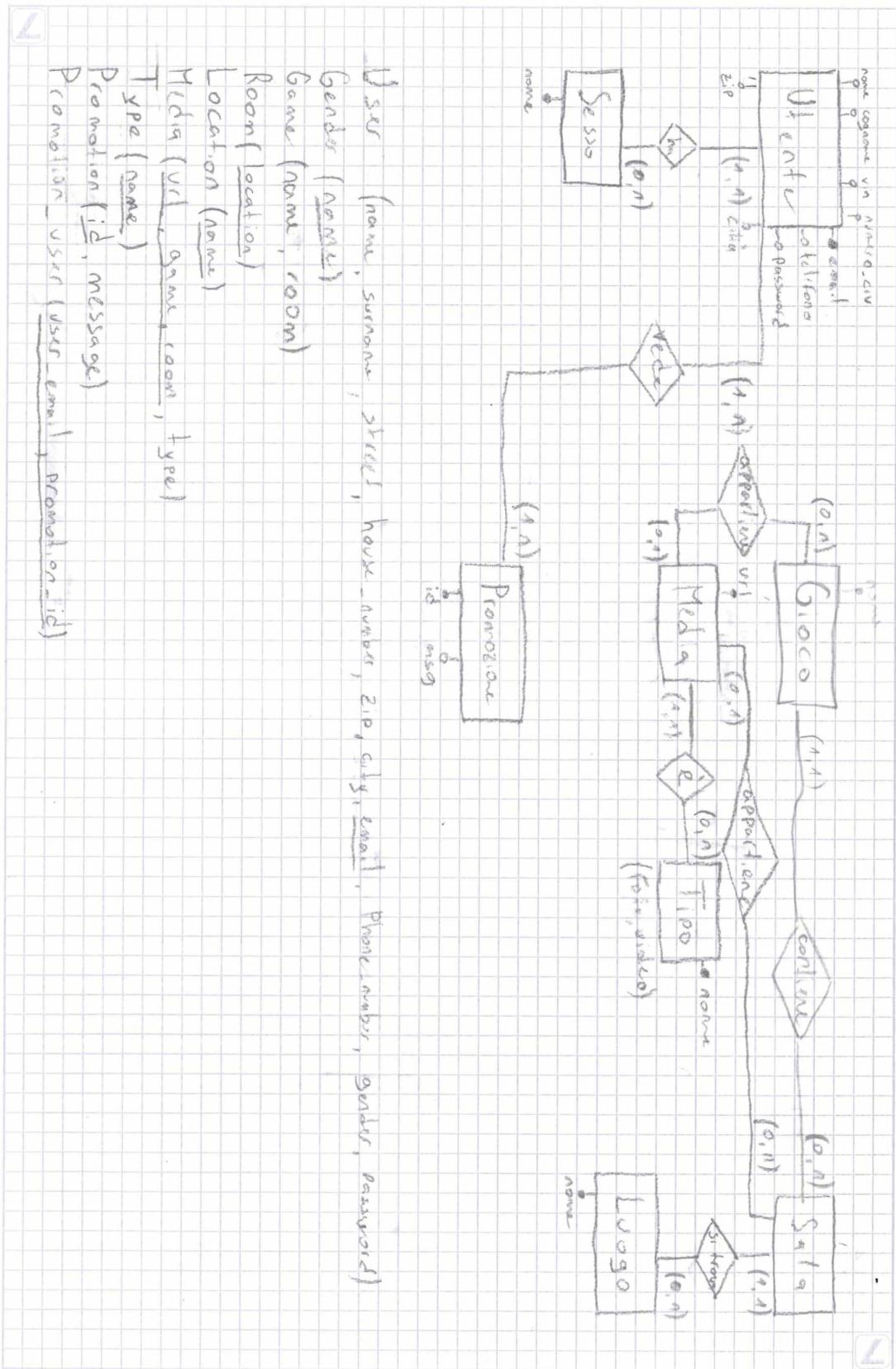
GESTIONE CASINÒ | Diario di lavoro - 22.02.2019

Matan Davidi, Thor Düblin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 22 Febbraio 2019

Lavori svolti

Durante la giornata di oggi Matan ha terminato lo schema logico usato per la progettazione del database, che trovate di seguito.



Durante la giornata di oggi Thor ha continuato l'installazione del WebDriver di Selenium, riuscendo a risolvere il problema legato al pom.xml dell'ultima lezione. Mattia Toscanelli oggi ha continuato con l'adattamento del sito web al contesto. Più precisamente ha completato la pagina di registrazione inserendo tutti i vari input necessari ed in seguito ha completato la pagina di login inserendo un'immagine più inerente alla pagina e i inoltre l'opzione per recuperare la password in caso fosse stata dimenticata. Infine ha iniziato a creare la pagina di recupero della password e di conferma della email.

Oggi Carlo Pezzotti ha sviluppato la classe per gestire le connessioni al database, la classe funziona nel seguente modo: quando si vuole istanziare un nuovo

database bisogna passare come parametro le seguenti informazioni ([host],[porta],[nomeDatabase],[username],[password]):

```
private $db;

function __construct($host,$port,$dbname,$username,$password)
{
    $this->db = new PDO("mysql:host=$host;port=$port;dbname=$dbname", $username, $password);
}
```

i metodi che ho reputato di essere utili sono i seguenti:

- executeQuery --> metodo che data una query passata come parametro la esegue e ritorna il risultato

```
public function executeQuery($query){
    $result = $this->db->query($query);
    if ($result === FALSE) {
        throw new InvalidArgumentException("Failed to load schema is not exists or you are not permission");
    }
    return $result->fetch();
}
```

- printTableQuery --> metodo che data una query di select stampa una tabella html

```
public function printTableQuery($selectQuery){
    $result = $this->db->query($selectQuery);
    if ($result === FALSE) {
        throw new InvalidArgumentException("Failed to load schema is not exists or you are not permission");
    }
    echo "<table id='tbl'><tr>";
    while($row = $result->fetch())
    {
        echo "<tr>";
        for ($i=0;$i<(sizeof($row)-1);$i++)
        {
            echo "<td>".$row[$i]."</td>";
        }
        echo "</tr>";
    }
    echo "</table>";
}
```

- insertUser --> metodo che dato un utente passato da parametro lo inserisce nel database.

```
public function insertUser($user){
    if(gettype($user) == "object"){
        if(get_class($user) == "User"){
            $query = "Insert into users(id, nome, cognome) values(null,'". $user->getName()."','". $user->getSurname()."')";
            $this->executeQuery($query);
        }else{
            throw new InvalidArgumentException(get_class($user)." is not a User class");
        }
    }else{
        throw new InvalidArgumentException(gettype($user)." is not a User class");
    }
}
```

La classe di test non fa altro che testare la classe sia a livello globale che a livello modulare dei singoli metodi

```
public function testDatabaseConnect(): void
{
    $this->assertInstanceOf(Database::class,new Database("127.0.0.1",3306,"provaCasino","root","root"));
}
public function testCannotConnect(): void
{
    $this->expectException(PDOException::class);
    $this->assertInstanceOf(Database::class,new Database("127.0.0.2",3306,"provaCasino","root","root"));
}
public function testInserUser() : void{
    $db = new Database("127.0.0.1",3306,"provaCasino","root","root");
    $lastId = $db->executeQuery("select max(id) from users")[0];
    $db->insertUser(new User(
        "Carlo",
        "Pezzotti",
        "2000-12-01",
        "Capolago",
        "Via laveggio",
        9,
        "0788159957",
    ));
}
```

```

        "carlo.pezzotti@samtrevano.ch",
        "male",
        "Password&1",
        "Password&1"
    );
    $nowId = $db->executeQuery("select max(id) from users")[0];
    $this->assertTrue($lastId+1 == $nowId);
}

public function testCannotInsertUser():void{
    $this->expectException(InvalidArgumentException::class);
    $db = new Database("127.0.0.1",3306,"provaCasino","root","");
    $db->insertUser("prova");
}

```

Matteo oggi ha risolto il problema della scorsa lezione con l'aiuto di Carlo per la quale non si riusciva a collegare Jenkins a Github e quindi ha messo in comunicazione i due sistemi facendo sì che quando si esegue un push il sistema esegue i test di prova creati in precedenza da Carlo.
Inoltre ha iniziato a scrivere la documentazione sull'installazione e configurazione di Jenkins portando a termine la prima parte.

Problemi riscontrati e soluzioni adottate

Carlo ha sollevato il discorso dell'indice utilizzato per la tabella 'User' del database: secondo lui, infatti, sarebbe meglio aggiungere una colonna 'id' all'entity set perché questo renderebbe le operazioni in PHP più semplici e veloci, mentre Matan afferma che così facendo si aggiungerebbe una colonna inutile all'entità che appesantirebbe qualsiasi operazione su di essa.

Abbiamo intenzione di discuterne al prossimo stand-up.

Inoltre Matan ha avuto problemi a scannerizzare il foglio sul quale ha disegnato il diagramma E/R e lo schema logico sulla stampante di scuola. Alla fine ha dovuto salvare il risultato della scansione, sotto forma di file PDF, su una chiavetta USB, utilizzando il menù "Pannello USB".

Thor ha risolto il problema degli errori quando digitava "mvn clean install" nella cartella del progetto di Selenium. I problemi erano causati da un errata configurazione del file pom.xml, che non riconosceva delle versioni di Maven.

Il seguente codice è quello modificato, che permette il corretto funzionamento del "mvn clean install". Inoltre ha installato Eclipse.

Mattia Toscanelli ha avuto difficoltà ad accedere al server ftp in quanto aveva dimenticato i dati d'accesso. Inoltre ha avuto problemi nel mettere responsive l'input select. Dopo qualche modifica in css è riuscito.

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>MySel20Proj</groupId>
    <artifactId>MySel20Proj</artifactId>

```

Un altro problema riscontrato oggi è quello della configurazione proxy da parte di Maven, che non permetteva lo scaricamento da internet, per risolvere questo problema, è bastato andare nella cartella conf di Maven ed aggiungere il codice per la configurazione del Proxy.

```

<proxy>
    <id>cpt.local</id>
    <active>true</active>
    <protocol>http</protocol>
    <host>proxy.cpt.local</host>
    <port>8080</port>
    <username>proxyuser</username>
    <password>somepassword</password>
</proxy>

```

Il problema nel collegamento era dovuto alle chiavi SSH e non è stato risolto ma semplicemente aggirato utilizzando al posto delle chiavi un webhook dato che la repository di github è pubblica e non necessita quindi di un'autenticazione. Il webhook creato è il seguente:

Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

http://134.209.13.21:8080/github-webhook/

Content type

application/json

Secret

Which events would you like to trigger this webhook?

- Just the push event.
- Send me everything.
- Let me select individual events.

Active

We will deliver event details when this hook is triggered.

[Update webhook](#)

[Delete webhook](#)

In seguito a ciò abbiamo riscontrato un altro problema che è quello per cui non vi è un'impostazione propria di GitHub nei WebHook per eseguire il trigger dopo un commit e quindi Carlo ha iniziato a scrivere uno script che esegua questo compito.

Punto della situazione rispetto alla pianificazione

Siamo in orario con la pianificazione

Programma di massima per la prossima giornata di lavoro

Matan deve preoccuparsi di mostrare lo schema logico al cliente in modo da poter correggere eventuali errori.

Mattia continuerà la pagina di password smarrita e la pagina di verifica della email.

Carlo Pezzotti si occuperà di fare una classe che si occuperà di mettere insieme tutte le classi sviluppate fin ora

Matteo la prossima lezione vuole riuscire a far funzionare correttamente GitHub con Jenkins e terminare la documentazione riguardante l'installazione e configurazione di Jenkins.

GESTIONE CASINÒ | Diario di lavoro - 27.02.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 27 Febbraio 2019

Lavori svolti

Matan ha cominciato a implementare una prima bozza di database MySQL seguendo lo schema logico terminato nell'ultima lezione. Il risultato è un database "cashyland" contenente le seguenti tabelle:

- game(name, room)
- gender(name)
- location(name)
- media(url, game, room, type)
- promotion(id, message)
- promotion_user(user_email, promotion_id)
- room(location)
- type(name)
- user(name, surname, street, house_number, zip_code, city, email, phone_number, gender, password, verified)

Thor ha continuato la configurazione del WebDriver di Selenium, in questa lezione ha importato il progetto di prova in Eclipse e installato le plugin m2eclipse, successivamente ha provato a abilitare la gestione delle dipendenze del progetto, tuttavia il campo che permette questa operazione non risulta presente.

Matteo oggi ha modificato innanzitutto le impostazioni di sicurezza di Jenkins dato che ci siamo resi conto di non avere ancora un login per accedere alla dashboard del software. Dopodiché ha continuato con la documentazione sulla configurazione di Jenkins terminando la parte di configurazione iniziale e di creazione di un nuovo elemento.

Carlo Pezzotti oggi ha sviluppato un mail sender per l'autenticazione dell'account. Per l'invio delle mail ho utilizzato una libreria composer chiamata PHPMailer trovata su github <https://github.com/PHPMailer/PHPMailer>. Ho deciso di dividere il lavoro in due file. Uno che invia le mail e un che grazie a un parametro get riesce ad andare ad attivare l'utente all'interno del database. Durante lo stand up è uscito fuori che dovevamo aggiungere una nuova colonna alla tabella che contiene gli utenti ovvero un flag booleano che verifica se un utente è stato verificato oppure no.

Mattia Toscanelli come per annunciato nel diario scorso ha sviluppato la pagina HTML di password smarrita e la pagina HTML per la verifica della mail. Durante la lezione si è assentato per svolgere la presentazione inerente al suo progetto scorso.

Il codice importante di oggi è il seguente:

```
public function mailSend($email){  
    $this->mail->addAddress($email); // Name is optional  
    $this->mail->isHTML(true); // Set email format to HTML  
    $this->mail->Subject = 'Hi there! Verify your account!';  
    $cryptedMail = $email ^ $this->privateKey;  
    $byte = "";  
    for ($i=0; $i < strlen($cryptedMail); $i++) {  
        $byte .= ord($cryptedMail[$i]) . "-";  
    }  
    $this->mail->Body = 'How are you? \r\n this is your link: http://cashyland.tk/validate.php?id='.$byte;  
  
    if(!$this->mail->send()) {  
        echo 'Message could not be sent.';  
        echo 'Mailer Error: ' . $this->mail->ErrorInfo;  
    } else {  
        echo 'Message has been sent';  
    }  
}
```

la seguente funzione serve per inviare una mail con all'interno il link per confermare l'account. La conferma ho deciso di gestirla utilizzando l'email dell'utente dato che all'interno del database è univoca. Quando invio la mail cripto la mail con una key privata stabilita oggi con i compagni durante la lezione (SGG<?rpF3FTebqx?(kgQR:hsq'mqZIVH).

Al momento dell'invio leggo i byte generati dall'encoding della mail e li separo con un "-" per poterli leggerli nella pagina di validazione.

Problemi riscontrati e soluzioni adottate

Il docente/cliente era assente durante la giornata di oggi, quindi Matan non ha potuto mostrargli lo schema logico finito. Ha quindi dovuto cominciare a implementare il database temporaneamente, prendendo per scontato che lo schema andasse bene. Eventualmente dovrà modificarlo o ricrearlo completamente in caso non andasse bene.

Siccome "Enable Dependency Management", operazione mostrata nella documentazione di Selenium per importare il progetto, non era presente, si è documentato sul come risolvere questo problema, senza trovare soluzione.

Carlo Pezzotti all'inizio non riuscivo a trovare un modo per poter inviare le email con PHP. In seguito ho trovato un modo utilizzando il metodo "mail" fornito da php però non riuscivo a capire il funzionamento, quindi mi sono munito di una libreria composer esterna che si connette con il protocollo smtp sulla porta 25 con le credenziali di autenticazione, per inviare le email.

Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario

Programma di massima per la prossima giornata di lavoro

Matan deve preoccuparsi di mostrare lo schema logico al cliente in modo da poter correggere eventuali errori.

Thor deve abilitare la gestione delle dipendenze del progetto Selenium di prova.

Matteo vorrebbe terminare la documentazione su Jenkins e revisionarla per verificare che sia scritta bene così da non doverci più pensare.

Carlo svolgerà una pagina php per poter autenticare un utente.

Mattia Toscanelli inizierà e finirà la pagina di recupero password ed inoltre inizierà la pagina di benvenuto.

GESTIONE CASINÒ | Diario di lavoro - 01.03.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 1 marzo 2019

Lavori svolti

Oggi Matan ha, come prima cosa, finalmente mostrato lo schema logico del database al cliente, che ha confermato che andasse bene. Quindi non c'è stato niente da aggiungere, togliere o modificare al database creato la lezione scorsa, che rimane definitivamente così:

- game(name, room)
- gender(name)
- location(name)
- media(url, game, room, type)
- promotion(id, message)
- promotion_user(user_email, promotion_id)
- room(location)
- type(name)
- user(name, surname, street, house_number, zip_code, city, email, phone_number, gender, password, verified)

In seguito ha aiutato Carlo con la creazione del database sul server DigitalOcean, dato il problema con lo script esportato da MySQL Workbench.

Infine ha documentato il database nel capitolo 2.2, Design dei dati e database, spiegando gli schemi realizzati e la struttura delle tabelle.

Oggi Matteo ha terminato la documentazione sull'installazione e configurazione di Jenkins con il relativo collegamento con GitHub. Inoltre ha creato una nuova repo di GitHub da utilizzare come cartella di produzione sulla quale Jenkins dopo i test dovrà pushare ma non è riuscito a terminare lo script che esegue questo lavoro. Lo script realizzato al momento è il seguente.

```
cp -r ./GestioneCasino/. ./GestioneCasinoProduction/
cd GestioneCasinoProduction
git add .
git commit -m "Jenkins commit"
git push https://JenkinsCasino:[REDACTED]@github/JenkinsCasino/GestioneCasinoProduction.git HEAD:master
```

Oggi Thor ha fatto funzionare un paio di test di esempio utilizzando il WebDriver di Selenium, e ha risolto il problema legato ai plugin di m2eclipse, che aveva riscontrato l'ultima lezione.

Ha visto come implementare appunto i comandi che venivano eseguiti dai test nel linguaggio di programmazione di java con Eclipse.

Problemi riscontrati e soluzioni adottate

Carlo ha avuto problemi con l'importazione del file dump del database creato da Matan durante l'ultima lezione: infatti il file è stato esportato da un database MySQL ed il tentativo di importazione è stato fatto all'interno di un database MariaDB, quindi è fallito più volte. Alla fine Matan l'ha aiutato inserendo alcune delle tabelle in ordine in modo che la creazione dei vincoli di integrità referenziale (foreign key) non desse problemi.

Inoltre, Carlo ha avuto problemi con l'accesso remoto al server MariaDB presente sul server DigitalOcean, che ha risolto seguendo una [guida](#) che diceva di commentare la riga `-address = 127.0.0.1` nel file di configurazione di MySQL.

Il problema riscontrato da Matteo con lo script da far eseguire a Jenkins è che il push richiede le credenziali e anche se esse vengono passate, come si può vedere dalla foto sopra, lo script non viene portato a termine perché vi è un errore con le credenziali. La soluzione al problema non è ancora nota.

Thor ha risolto il problema legato ai plugin di m2eclipse, importando correttamente il percorso dal sito <http://download.eclipse.org/technology/m2e/releases>, che nella scorsa lezione non corrispondeva alla versione utilizzata di Selenium riscontrando errori.

Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

Programma di massima per la prossima giornata di lavoro

Matteo la prossima lezione vuole completare lo script così da collegare finalmente GitHub con Jenkins in maniera corretta e perfettamente funzionante.

GESTIONE CASINÒ | Diario di lavoro - 13.03.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 13 marzo 2019

Lavori svolti

Durante la giornata di oggi, Matan ha dovuto aggiungere una colonna alla tabella user del database dove deve venir inserita la data e l'ora della registrazione di un utente in modo da poter usare questo dato per cancellare eventuali utenti non confermati dopo un giorno dalla loro registrazione. Un evento pianificato per essere avviato ogni 12 ore si occupa di cancellare questi utenti. Inoltre ha dovuto creare un trigger che si accertasse che la data di registrazione non fosse nulla, in caso contrario deve essere inserita la data e l'ora dell'inserimento.

Questo può essere fatto con il seguente codice:

```
ALTER TABLE cashyland.user ADD COLUMN registration_date DATETIME;
USE cashyland;
DELIMITER //
CREATE EVENT IF NOT EXISTS delete_unverified_users
ON SCHEDULE EVERY 12 HOUR
DO
BEGIN
    DELETE FROM cashyland.user WHERE verified = 0 AND registration_date <= subdate(NOW(), 1);
END
//
```

Invece il trigger viene creato con il codice seguente:

```
USE cashyland;
DELIMITER //
CREATE TRIGGER before_user_insert BEFORE INSERT ON user
FOR EACH ROW
BEGIN
    IF NEW.registration_date IS NULL THEN
        SET NEW.registration_date = NOW();
    END IF;
END
//
```

Oggi Matteo ha provato a rendere funzionante il collegamento fra github e Jenkins facendo sì che quest'ultimo eseguisse i push delle nostre modifiche del codice sulla repository di produzione. Questo ha però creato ancora problemi.

Oggi Carlo si è occupato di sistemare il codice scritto nelle scorse giornate di lavoro rendendolo ottimizzato. Inoltre si è occupato di gestire il reset della password quando viene richiesto. Infine ho generalizzato delle classi che prima ritenevo non generalizzate e ho aggiunto in file php che si occupa di gestire i vari require e di inizializzare le variabili necessarie

Il codice più importante è il seguente:

->File php per cambiare la password

```
require "php/loader.php";
if (isset($_GET['id'])) {
    $id = $_GET['id'];
    $email = urldecode($id);
    $email = $email ^ $privateKey;

    if(!($db->existsUserByEmail($email)) == "boolean")){
        if($_SERVER['REQUEST_METHOD'] == "POST"){
            if(isset($_POST["password"]) && isset($_POST["repassword"])){
                $db->executeQuery('update user set password = "'.$_POST["password"].'" where email = "'.$email.''");
            }
        }else{
            echo"
            <body>
                <form action='changePassword.php?id=".urlencode($email^$privateKey)."' method='post'>
                    <span>Password:</span>
                    <input type='password' name='password'><br>
                    <span>Repeat-Password:</span>
                    <input type='password' name='repassword'><br>
                    <input type='submit' name='VAI!' value='VAI!'>
                </form>
            </body>
        ";
    }
}
```

```

        </body>
    ";
}
}else{
    echo "Qualcosa è andato storto :(";
}
}
}

```

Il seguente codice permette ad utente di cambiare la propria password. Per la gestione dell'email uso il metodo utilizzato in precedenza per verificare l'utente.

->File php che carica tutto il necessario

```

require "database/database.php";
require "sendMail/sendMail.php";
require "user/user.php";
$GLOBALS['db'] = new Database("127.0.0.1",3306,"cashyland","casinoAdmin","Casin02018");
$GLOBALS['mailSender'] = new SendMail();
$GLOBALS['privateKey'] = "SGG<?rpF3FTebqx?{kgQR:hsq'mqZ!VH";

```

Il seguente codice serve per caricare tutto ciò che penso sia necessario.

Oggi Thor ha creato un programma che fa i test grafici tra la navigazione delle pagine di Login e di Registrazione del sito web di Cashy Land, alla schermata di Login i test automatizzati controllano che se viene premuto il pulsante REGISTRATI esso porterà alla pagina di registrazione. Mentre alla pagina di registrazione, premendo il pulsante ACCEDI nella barra di navigazione esso porterà alla pagina di Login. Successivamente si vedono le linee di codice che permettono la pressione del pulsante nel test automatizzato.

```

driver.findElement(By.name("login")).submit();
driver.findElement(By.name("register")).submit();

```

E un pezzo di codice che gestisce l'attesa di navigazione tra una pagina e l'altra, assicurandosi che la navigazione tra una pagina e l'altra non impieghi troppo tempo, altrimenti riproverà la navigazione.

```

(new WebDriverWait(driver, 5)).until(new ExpectedCondition<Boolean>() {
    public Boolean apply(WebDriver d) {
        return d.getTitle().toLowerCase().equals("GestioneCasino - Registrazione");
    }
});

```

Mattia Toscanelli oggi si è occupato del controllo lato client del form di registrazione. Non ha avuto particolari problemi se non con l'implementazione della libreria "notify.js" che permette di ricevere una notifica in caso ci fosse un errore in uno dei campi. L'errore è stato risolto mettendo in un determinato ordine i riferimenti

```

<script src="js/jquery.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/jquery.parallax.js"></script>
<script src="js/custom.js"></script>
<script src="js/notify.js"></script>
<script src="js/registration.js"></script>

```

Problemi riscontrati e soluzioni adottate

Matteo ha riscontrato, come la settimana scorsa, dei problemi con le credenziali e con le versioni di Github ed ha provato a risolverli aggiungendo una chiave ssh all'utente GitHub.

 **gruppocasin02018@hotmail.com**
b1:04:97:47:9c:a1:6c:1f:91:db:2f:9a:c9:e4:56:88
Added on 13 Mar 2019 Last used within the last week — Read/write Delete

Questa operazione non è però risultata efficace e dunque ha provato cambiando lo script aggiungendo un pull prima delle altre operazioni di git così da lavorare sempre sulla versione aggiornata anche se teoricamente l'utente di Jenkins sarà l'unico che utilizzerà quella repo. Questa soluzione ha solamente modificato l'errore ottenuto nel seguente.

```
! [rejected]      HEAD -> master (non-fast-forward)
error: failed to push some refs to 'https://JenkinsCasino:Casin02018@github.com/samtcasino/GestioneCasi
noProduction.git'
hint: Updates were rejected because a pushed branch tip is behind its remote
hint: counterpart. Check out this branch and integrate the remote changes
hint: (e.g. 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

Programma di massima per la prossima giornata di lavoro

Matteo la prossima settimana proverà a chiedere aiuto per risolvere i suoi problemi che calcolava di risolvere entro oggi.

Thor la prossima lezione testerà il funzionamento dei test tra una pagina e l'altra.

GESTIONE CASINÒ | Diario di lavoro - 15.03.2019

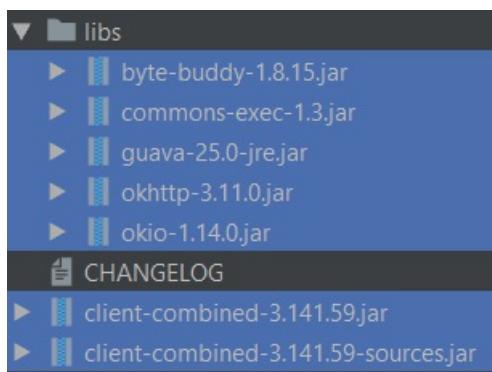
Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 15 marzo 2019

Lavori svolti

Data l'assenza di Thor, oggi Matan si è occupato di prendere in mano il suo lavoro e portarlo avanti. Come prima cosa, quindi, si è informato velocemente su come funzionasse Selenium, poi ha creato un'applicazione di test usando la libreria di Selenium WebDriver. L'applicazione che ha creato stampa su terminale il titolo della pagina del sito del casinò (<http://www.cashyland.tk>). Per farlo si è aiutato con questo [tutorial per creare i test](#) (<https://wiki.saucelabs.com>) e con quest'altro [tutorial per avviare i test senza interfaccia grafica](#) (<https://www.built.io/blog/run-selenium-tests-in-headless-browser>).

Come prima cosa ha dovuto scaricare le librerie di Selenium WebDriver seguendo questo [link](#) (<https://www.seleniumhq.org/download/>), poi ha creato un progetto di prova per cominciare a fare qualche test con Selenium che ha chiamato `SeleniumTest`, all'interno del quale ha importato tutte le librerie scaricate:



In seguito ha scritto il seguente codice:

```
// System.setProperty("webdriver.chrome.driver", "ChromeDriverPath");
// ChromeOptions options = new ChromeOptions();
// options.addArguments("headless");
// options.addArguments("window-size=1200x600");
WebDriver driver = new ChromeDriver(/* options */);
driver.get("http://www.cashyland.tk");

WebElement title = driver.findElement(By.tagName("title"));
String titleText = title.getText();
System.out.println(titleText);

GroovyTestCase.assertEquals(titleText, "Neuron HTML CSS Template");

driver.quit();
```

Dove la parte commentata serve per eseguire il test senza interfaccia grafica. Questo snippet di codice apre la pagina della gestione del casinò, trova l'elemento "title", prende il testo e certifica che il testo è uguale a "Neuron HTML CSS Template".

Matteo ha risolto, probabilmente, il problema che non lo lasciava collegare Jenkins alla repo di produzione semplicemente salvando le credenziali nel sistema con il comando:

```
git config --global credential.helper store
```

e modificando lo script eliminando le parti inutili del comando push.

```
cp -rf ./GestioneCasino/. ./GestioneCasinoProduction/
cd ./GestioneCasinoProduction
git pull origin master
git add .
git commit -m "Jenkins commit"
git push
```

Il funzionamento dello script con Jenkins non è potuto però essere testato dato che esso non funziona come dovrebbe.

Mattia Toscanelli oggi ha lavorato ancora con la pagina di registrazione, in particolare con la verifica lato client dei dati. Questo è successo perché dopo qualche

test eseguito a mano si è scoperto che erano presenti degli errori. Uno degli errori riscontrati è il controllo delle due password, più precisamente la verifica che le due password siano uguali, di lunghezza minima 8, che abbia almeno un numero e infine che abbia almeno una maiuscola. Questo è stato risolto così:

```
//Controllo se le password sono uguali
function checkPassword(pas1, pas2){
    return (pas1 == pas2) && pas2.toLowerCase() != pas1 && /\d/.test(pas1) && pas1.length > 7;
}
```

Quando è presente un errore nel valore inserito all'interno di un campo, quest'ultimo diventa rosso. Però, quando vi si inserisce nuovamente un valore valido, quest'ultimo deve ritornare al colore di partenza. Questo non accadeva ed è stato corretto impostando il colore di sfondo del campo a bianco con il seguente codice:

```
//rimuove il colore rosso dagli input errati
function normal(input){
    input.style.backgroundColor = "white";
}
```

Infine ha corretto piccoli errori trascurabili e ha spostato la notifica di errore da in alto a destra ad in basso a sinistra con il seguente codice:

```
$.notify("Nome non valido!", { position: "bottom left" });
```

Carlo Pezzotti oggi ha cercato di implementare completamente la grafica con il backend, ma ho trovato alcuni problemi. Io e Mattia non ci siamo messi molto d'accordo quindi alcuni nomi non erano corretti tra di loro quindi risultava esserci errori, che non trovavo. Inoltre mi sono occupato di mettere a posto e completare tutti i test necessari ovvero, i test per la classe use, i test per la classe database e i test per la classe di spedizione delle mail. L'ultimo test di cui mi sono occupato è l'ultimo citato in precedenza:

```
final class SendMailTest extends TestCase
{
    public function testCanCreateSendMailObject(): void
    {
        $this->assertInstanceOf(SendMail::class,new SendMail());
    }

    public function testCanSendMail():void{
        $e = new SendMail();
        $this->assertTrue($e->mailSend("gruppoCasin0@hotmail.com","test","test"));
    }
}
```

Come descritto nei diari precedenti, per l'invio delle mail l'ho raggiunto grazie ad una libreria esterna.

Il metodo per l'invio ritorna true o false in base se una mail è stata spedita un modo corretto oppure no. Ciò l'ho utilizzato per la creazione del test.

Problemi riscontrati e soluzioni adottate

Mattia ha speso molto del suo tempo a mettere a posto un errore causato da un collega che non permetteva più di ritornare alla pagina precedente, indipendentemente da quale fosse, dalla pagina di registrazione, ma purtroppo non è riuscito a capire quale fosse né a sistemarlo, dunque è tornato sulla versione vecchia del codice e ha aggiunto le nuove modifiche descritte sopra.

Matteo, prima di trovare la soluzione spiegata sopra, ha provato a far funzionare lo script che esegue i push utilizzando su Jenkins come "post build operations" un git publisher ma esso esegue solamente il push sulla stessa repo in cui ha eseguito i test.

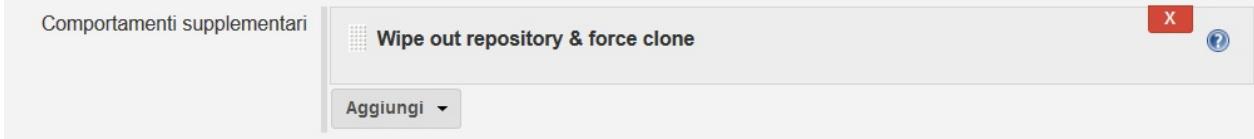
Fatto ciò ha riscontrato problemi con Jenkins, esso non esegue più le compilazioni correttamente dato che non riesce ad eseguire il fetch sulla cartella di GitHub.

```

Started by user GestioneCasino
Building in workspace /var/lib/jenkins/workspace/GestioneCasino
No credentials specified
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.GestioneCasino.url https://github.com/samtcasino/GestioneCasino.git # timeout=10
ERROR: Error fetching remote repo 'GestioneCasino'
hudson.plugins.git.GitException: Failed to fetch from https://github.com/samtcasino/GestioneCasino.git
    at hudson.plugins.git.GitSCM.fetchFrom(GitSCM.java:894)
    at hudson.plugins.git.GitSCM.retrieveChanges(GitSCM.java:1161)
    at hudson.plugins.git.GitSCM.checkout(GitSCM.java:1192)
    at hudson.scm.SCM.checkout(SCM.java:504)
    at hudson.model.AbstractProject.checkout(AbstractProject.java:1208)
    at hudson.model.AbstractBuild$AbstractBuildExecution.defaultCheckout(AbstractBuild.java:574)
    at jenkins.scm.SCMCheckoutStrategy.checkout(SCMCheckoutStrategy.java:86)
    at hudson.model.AbstractBuild$AbstractBuildExecution.run(AbstractBuild.java:499)
    at hudson.model.Run.execute(Run.java:1810)
    at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:43)
    at hudson.model.ResourceController.execute(ResourceController.java:97)
    at hudson.model.Executor.run(Executor.java:429)

```

Questo problema c'è da mercoledì 13 marzo ma non era stato notato. Non è stata ancora trovata una soluzione. È stato provato oggi a modificare inizialmente i permessi sul file git/config dato che uno dei messaggi di errore era dovuto ai privilegi sul file ma non è servito, così come cambiare il proprietario del file. Cercando su internet un'altra possibile causa del problema poteva essere la mancanza di spazio nella cartella /var/lib/jenkins/workspace ma pur avendo eliminato tutti i files inutili da essa la situazione non cambia. L'ultimo tentativo fatto, anche esso trovato su internet, è stato quello di aggiungere un comportamento supplementare a Jenkins così che ad ogni compilazione eliminasse la repo prima di forzare il clone ma ciò porta solo ulteriori errori.



Matan ha avuto il problema dove non viene trovata la classe usata per fare gli assert all'interno della sua applicazione, ossia GroovyTestCase, anche se essa è stata importata. Infatti viene mostrato il seguente errore:

```
Error:(22, 23) java: cannot access junit.framework.TestCase
  class file for junit.framework.TestCase not found
```

Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

Programma di massima per la prossima giornata di lavoro

Matteo ha intenzione di rimettere in funzione Jenkins e testare se lo script funziona con esso.

Mattia Toscanelli ha intenzione di creare la pagina di messaggio che è stata inviata una mail di recupero password ed inoltre inizierà la pagina di benvenuto.

Matan deve riuscire a sistemare il problema che presenta la sua applicazione a causa dell'importazione delle librerie.

GESTIONE CASINÒ | Diario di lavoro - 20.03.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 20 marzo 2019

Lavori svolti

Oggi Matan si è unito a Thor per cercare di far funzionare i test del front-end eseguiti con Selenium.

Come prima cosa Matan ha dovuto installare Maven seguendo questo [tutorial](https://maven.apache.org/install.html) (<https://maven.apache.org/install.html>), basta scaricarlo e aggiungere il percorso verso la cartella "bin" all'interno dell'archivio appena estratto alla variabile d'ambiente PATH.

Ha poi creato un nuovo progetto SeleniumTest perché ha preferito cominciare da zero, quindi ha dovuto aggiungere un file pom.xml alla root del progetto con il seguente contenuto:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>SeleniumTest</groupId>
<artifactId>SeleniumTest</artifactId>
<version>1.0</version>
<dependencies>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-server</artifactId>
        <version>3.0.1</version>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter</artifactId>
        <version>5.5.0-M1</version>
    </dependency>
</dependencies>
</project>
```

In seguito ha dovuto creare un file al seguente percorso:

```
"C:\Users\<utente>\.m2\settings.xml"
```

Contenente le impostazioni per la proxy della scuola:

```
<settings>
<proxies>
    <proxy>
        <id>cpt</id>
        <active>true</active>
        <protocol>http</protocol>
        <host>10.20.0.1</host>
        <port>8080</port>
        <username>nomeUtente</username>
        <password>password</password>
    </proxy>
</proxies>
</settings>
```

In seguito ha aperto una finestra di terminale e si è diretto nella root del progetto SeleniumTest e ha digitato il comando:

```
mvn clean install
```

In seguito Matan ha scaricato e aggiunto alla variabile d'ambiente PATH il WebDriver [ChromeDriver](http://chromedriver.chromium.org/) (<http://chromedriver.chromium.org/>)

```
System.setProperty("webdriver.chrome.driver", "D:\\Programmi\\ChromeDriver\\chromedriver.exe");
ChromeOptions options = new ChromeOptions();
options.addArguments("headless");
options.addArguments("window-size=1200x600");

WebDriver driver = new ChromeDriver();
driver.get("http://www.cashyland.tk");

String titleText = driver.getTitle();
```

```
System.out.println(titleText);
driver.quit();
```

Mattia ha verificato se la pagina di registrazione fatta la scorsa lezione fosse funzionante in vari casi. Tutti i test manuali che ha eseguito hanno avuto successo positivo.

In seguito ha creato la pagina di reset della password. La struttura della pagina è molto simile alle altre della registrazione, ma quest'ultima presenta solamente due input di tipo password.



Cambia la tua vecchia password

Inserisci due volte la tua nuova password. (min. 8 caratteri, Maiuscola e Numero)

Password:

 Password

Ripeti Password:

 Ripeti Password

REGISTRATI

In questa pagina ha fatto pure il controllo lato client delle password, questo è stato fatto così:

```
//Controllo se le password sono uguali
function checkPassword(pas1, pas2){
    return (pas1 == pas2) && pas2.toLowerCase() != pas1 && /\d/.test(pas1) && pas1.length > 7;
}

//Controllo lato client dei campi con eventuali notifiche in caso di errore.
function checkAll(){
    var inputs = document.getElementsByTagName("input");
    if(checkPassword(inputs[0].value, inputs[1].value)){
        document.getElementById("registration_form").submit();
    }else{
        if(!checkPassword(inputs[0].value, inputs[1].value)){
            inputs[0].style.backgroundColor = "#ffcccc";
            inputs[1].style.backgroundColor = "#ffcccc";
            $.notify("Password non valida o non corrispondente!", { position:"bottom left" });
        }else{
            inputs[0].style.backgroundColor = "white";
            inputs[1].style.backgroundColor = "white";
        }
    }
}
```

Infine ha incominciato a modificare la pagina di Benvenuto.

Oggi Carlo ha cercato per tutta la lezione di installare un mail server su ubuntu 18.04 ma senza riuscirci. Non ho riscontrato errori, però non ho proprio capito perché il server mail non andasse.

Oggi Thor ha spiegato a Matan come utilizzare il WebDriver di Selenium, indirizzandolo verso la strada corretta per non fargli perdere troppo tempo, successivamente si è occupato di creare il codice Java per il WebDriver di Selenium, che si occupa di navigare dalla Pagina di Registrazione, a quella di Login, a quella per resettare la password.

Il seguente codice è quello che dovrebbe gestire la navigazione tra una pagina e l'altra.

```
(new WebDriverWait(ct.driver, 10)).until(new ExpectedCondition<Boolean>() {
    public Boolean apply(WebDriver d) {
        return d.getTitle().startsWith("gestionecasino - login");
    }
});
```

Matteo ha messo a posto lo script che esegue il push su git aggiungendo una riga che setta ogni volta che si fa la copia dei files il remote url dato che esso veniva cambiato dopo questa operazione. Inoltre è stata modificata la riga dove si copiano i files facendo sì che essa non copi le cartelle nascoste con tutti i files di configurazione di git che poi andrebbero in conflitto. L'ultima modifica effettuata allo script è stata quella di consentire il merge delle unrelated histories dato che esse generavano un errore.

```
cd GestioneCasino
git pull
cd ..
rsync -a --exclude=".git" ./GestioneCasino/. ./GestioneCasinoProduction/
cd ./GestioneCasinoProduction
git remote set-url origin https://github.com/samtcasino/GestioneCasinoProduction.git
git pull origin master --allow-unrelated-histories
git add .
git commit -m "Jenkins commit"
git push
```

Fatto ciò ha modificato i permessi sulle cartelle di git dando più privilegi all'utente di Jenkins così che esso potesse fare tutte le operazioni desiderate.

Problemi riscontrati e soluzioni adottate

Installazione del mail server

Thor ha installato il ChromeDriver che permette di gestire chrome tramite un bot, problema che è stato riscontrato l'ultima lezione e risolto.

In più in questa lezione si sono riscontrati degli errori di navigazione tra una pagina e l'altra, dovuto al tempo troppo lungo per navigare da una pagina all'altra.

Matteo ha riscontrato alcuni problemi con i permessi sulla cartella nascosta .git che contiene i files di git. Essa non consentiva all'utente di Jenkins di copiarla. Impostando come possessore della cartella l'utente desiderato il problema si è risolto.

Inoltre poco prima della pausa Jenkins ha smesso di funzionare impedendo qualsiasi collegamento al suo pannello di controllo. Dopo varie ricerche inutili e svariato tempo perso ad aspettare che ripartisse (dato che si pensava fosse un problema di velocità di connessione) il problema è stato risolto semplicemente eseguendo il reboot del server.

Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

Programma di massima per la prossima giornata di lavoro

Installare un framework per i test da utilizzare in parallelo a Selenium.

Installare un mail server.

GESTIONE CASINÒ | Diario di lavoro - 22.03.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 22 marzo 2019

Lavori svolti

È stato deciso che non verrà installato un mail server sul server remoto di produzione perché risulta più semplice utilizzare la libreria "PHPMailer" [PHPMailer](https://github.com/PHPMailer/PHPMailer) (<https://github.com/PHPMailer/PHPMailer>) utilizzando come server di posta elettronica Hotmail tramite l'indirizzo email "gruppoCasin02018@hotmail.com".

Matan oggi ha ripreso la documentazione, scrivendo l'abstract all'interno del capitolo 1.2, Abstract.

Mattia Toscanelli ha cercato delle immagini da aggiungere al sito, senza diritti d'autore. Il sito nel quale le ha trovate è il seguente:
<https://www.pexels.com/search/casino/>

Oggi Carlo e Mattia hanno lavorato insieme per mettere insieme al meglio la part front-end e back-end. Siamo riusciti con successo a fare ciò. Carlo aveva bisogno di sapere come fosse la struttura per gestire gli errori e Mattia mi ha fatto notare che nel progetto esiste una libreria che permette di notificare. Alla fine siamo giunti alla conclusione che è meglio utilizzare questo approccio.

Per passare il testo di errore alla notifica uso lo stesso approccio descritto nel diario scorso, ovvero di salvare l'errore su un cookie e mostrarlo in seguito, grazie al seguente metodo:

```
function getCookie(cname) {
    console.log("Loo");
    var name = cname + "=";
    var decodedCookie = decodeURIComponent(document.cookie);
    var ca = decodedCookie.split(';');
    for(var i = 0; i < ca.length; i++) {
        var c = ca[i];
        while (c.charAt(0) == ' ') {
            c = c.substring(1);
        }
        if (c.indexOf(name) == 0) {
            var returnValue = c.substring(name.length, c.length);
            while(returnValue.includes("+")){
                returnValue = returnValue.replace("+", " ");
            }
            console.log(returnValue);
            return returnValue;
        }
    }
    return "";
}

if(getCookie("error") != ""){
    var inputs = document.getElementsByTagName("input");
    inputs[0].style.backgroundColor = "#ffcccc";
    inputs[1].style.backgroundColor = "#ffcccc";
    $.notify(getCookie("error"), { position:"bottom left" });
} else{
    inputs[0].style.backgroundColor = "white";
    inputs[1].style.backgroundColor = "white";
}
```

La seguente funzione in base al nome del cookie ritorna il suo valore.

La vita del cookie è molto breve, infatti abbiamo deciso di farlo vivere solo 1 secondo.

Oggi Matteo ha risolto tutti i problemi di permessi che l'utente di jenkins riscontrava con tutte le cartelle di git ed i relativi files di configurazione.

Ha poi inserito lo script come Post Build Task nel lavoro di jenkins ed ha eseguito dei test di funzionamento che non hanno ritornato risultati positivi.

Oggi Thor ha continuato la creazione del codice Java del WebDriver di Selenium, che testa la navigazione tra le pagine di Registrazione, Login e Smarrimento passw ord, il problema sulla quale si è concentrato oggi è il come gestire l'attesa tra una pagina e l'altra prima di eseguire una qualsiasi iterazione con essa. Per questo sono stati provati vari metodi che non hanno prodotto un risultato positivo.

Problemi riscontrati e soluzioni adottate

Matteo ha riscontrato inizialmente molti problemi di permessi che sono stati tutti risolti cambiando proprietario o permessi della cartella o file in questione. Dopo di che inserendo lo script come Post Build Task di jenkins ha notato che esso non viene mai eseguito perché vi è sempre un messaggio che dice **Logical operation result is FALSE** anche se tutto viene eseguito senza alcun problema.

```

OK (7 tests, 7 assertions)
Performing Post build task...
Could not match :build : False
Logical operation result is FALSE
Skipping script : cd /var/lib/jenkins/workspace
cd GestioneCasino
git pull
cd ..
rsync -a --exclude=".*" ./GestioneCasino/. ./GestioneCasinoProduction/
cd ./GestioneCasinoProduction
git remote set-url origin https://github.com/samtcasino/GestioneCasinoProduction.git
git pull origin master --allow-unrelated-histories
git add .
git commit -m "Jenkins commit"
git push

END OF POST BUILD TASK : 0
Finished: SUCCESS

```

Thor ha provato, insieme all'aiuto di Matan un paio di metodi per gestire l'attesa di caricamento tra una pagina e l'altra, i seguenti metodi dovrebbero attendere che la pagina si sia caricata completamente prima di eseguire una qualsiasi altra operazione su di essa.

Con il seguente metodo si attende che lo stato della pagina ritorni "complete", e che quindi la pagina si sia caricata completamente, tuttavia sembra che la pagina ritorni lo stato "complete" quando in realtà si trova ancora in una fase di caricamento.

```

ct.driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
new WebDriverWait(ct.driver, 10).until(
    driver -> ((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete"));

```

Anche in questo caso si aspetta che l'elemento il link Accedi sia accessibile prima di eseguire una qualsiasi azione, tuttavia per qualche motivo anche qui sembra che il link sia cliccabile quando in realtà non risulta così.

```

```Java
WebDriverWait w ait = new WebDriverWait(ct.driver, 60);
w ait.until(ExpectedConditions.elementToBeClickable(By.linkText("Accedi")));

```

Con il seguente metodo ci si aspetta che il titolo della pagina sia uguale a quello che ci si aspetta, tuttavia quando la pagina sta ancora caricando il titolo è diverso.

```

Java
(new WebDriverWait(ct.driver, 10)).until(new ExpectedCondition() {
public Boolean apply(WebDriver d) {
return d.getTitle().startsWith("gestionecasino - login");
}
});

```

Questi metodi ritornano tutti l'errore:

unknown error: Element ... is not clickable at point (781, 43). Other element would receive the click.

Che indica che si sta provando ad eseguire il comando per cliccare al momento in cui la pagina non si è ancora caricata.

## Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

## Programma di massima per la prossima giornata di lavoro

Matteo ha intenzione di far sì che jenkins esegua lo script.

Thor per la prossima giornata di lavoro ha intenzione di concludere i test che riguardano la navigazione tra pagine e cercare un framework per i protocolli di test.

# GESTIONE CASINÒ | Diario di lavoro - 27.03.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 27 marzo 2019

## Lavori svolti

Matan ha proseguito con la documentazione di progetto cominciando a documentare il capitolo 1.6, Planificazione.

Inoltre Matan ha cercato di aiutare Thor cominciando a informarsi su come verificare che le email di conferma di registrazione vengano mandate e arrivino correttamente. Questo può essere fatto grazie all'API della Oracle JavaMail (<https://javaee.github.io/javamail/>), che permette di scrivere del codice Java che si collega a un server IMAP e legge le sue email grazie al seguente codice:

```
try {
 Properties props = new Properties();
 //set email protocol to IMAP
 props.put("mail.store.protocol", "imaps");
 //set up the session
 Session session = Session.getInstance(props);
 Store store = session.getStore("imaps");

 try {
 //Connect to your email account
 store.connect("imap.hotmail.it", "gruppoCasin02018@hotmail.com", "fAKEPAsSwOrD");
 //Get reference to your INBOX
 Folder folder = store.getFolder("INBOX");
 //Open the folder in READ MODE only
 folder.open(Folder.READ_ONLY);
 //Get all the messages in INBOX into Message array
 Message[] messages = folder.getMessages();

 //Loop through all the messages in your INBOX
 for (Message msg : messages) {
 //Fetch the subject and content into string variables
 String mailSubject = (String) msg.getSubject();

 try {
 String mailContent = (String) msg.getContent();

 //print the subject and content on console
 System.out.println(mailSubject);
 System.out.println(mailContent);
 } catch (IOException ioe) {
 System.out.println(ioe.getMessage());
 }
 }
 } catch (MessagingException me) {
 System.out.println(me.getMessage());
 }
}
}
}
}
}
}
}
}
}
}
```

Oggi Matteo e Carlo hanno lavorato insieme per far funzionare i push di Jenkins, i problemi iniziali erano ancora, come la settimana scorsa, riguardanti i permessi e abbiamo provato a risolverli impostando per l'utente jenkins a tutte le cartelle full control con il seguente comando.

```
chmod -R 777 GestioneCasino
chmod -R 777 GestioneCasinoProduction
```

Questo comando salta però alcune cartelle, secondo noi a caso, e quindi abbiamo iniziato ad impostare i permessi a mano ma ci siamo resi conto che ad ogni cartella modificata ne appariva un'altra con i permessi mancanti ed abbiamo deciso quindi di provare a mettere tutti i comandi che facevamo eseguire a Jenkins in uno script bash e fargli eseguire solo quello. Alcuni problemi si sono risolti ma non tutti quindi abbiamo provato a cambiare l'utente utilizzato da Jenkins. Per fare ciò abbiamo cercato la soluzione su internet e la abbiamo trovata a questo link: <https://stackoverflow.com/questions/29926773/run-shell-command-in-jenkins-as-root>

`user` ma facendo ciò il sito di jenkins riscontrava dei problemi. Esso infatti non si apriva più e se si caricava non funzionava il CSS. Abbiamo quindi deciso di ritornare alla situazione precedente modificando nuovamente il file di configurazione e i permessi sulle cartelle. Fatto ciò improvvisamente le compilazioni hanno iniziato a funzionare senza un motivo effettivo e nessuno riesce a spiegarsene il motivo.

Abbiamo quindi fatto ulteriori test per controllare se non fosse un caso e tutti sono riusciti senza problemi.

Abbiamo quindi pulito la cartella GestioneCasinoProduction eliminando files inutili.

Oggi Thor ha cercato un framework su cui fare i test di selenium con eclipse dove alla fine è stato scelto JUnit, successivamente è stata completato il programma che si occupa di testare la pagina di benvenuto, e tutte le navigazioni tra le pagine possibili.

Inoltre ha iniziato a programmare i test che si occupano della registrazione degli utenti, iniziando a creare un file CSV che conterrà tutte le credenziali di più ipotetici utenti che proveranno a registrarsi (correttamente e non).

Oggi Mattia ha continuato con la grafica delle pagine del sito, più precisamente ha finito la pagina principale. Discutendo con Carlo ha pensato che essendo che deve sviluppare molte pagine web di scaricare un editor simile a Bootstrap Studio ma che sia gratis e ha trovato questo: <https://pinegrow.com>. Durante questa sessione si è dedicato ad imparare un po' la base su come si utilizza questa applicazione, ma non è ancora sicuro di utilizzarla. In conclusione non ha prodotto molto per quanto riguarda il progetto vero e proprio ma ha imparato ad utilizzare l'applicazione per poter "forse" velocizzare il suo lavoro.

## Problemi riscontrati e soluzioni adottate

Matan ha fatto fatica a trovare il materiale da scrivere nel capitolo 1.4 della documentazione, Analisi del dominio.

Matteo e Carlo dopo aver fatto funzionare le compilazioni hanno riscontrato che Jenkins crea tre cartelle che non si possono aprire e non ne conosciamo ancora il motivo.

I problemi riscontrati da Thor la scorsa lezione riguardo la navigazione tra le pagine di CashyLand è stata risolta con successo, si è scoperto che usando un Thread.sleep, esso aspettava che la pagina fosse completamente carica prima di eseguire l'operazione successiva, quindi si è optato per un Thread.sleep di un secondo (quanto basta per dare il tempo alla pagina di caricarsi completamente), successivamente un esempio funzionante di navigazione:

```
accedi = driver.findElement(By.linkText("Accedi")); //viene assegnato l'elemento
accedi.click(); //viene cliccato l'elemento
wai(); //metodo che fa una Thread.sleep di un secondo (con la gestione dell'eccezione)
System.out.println(driver.getTitle()); //viene stampato il titolo della pagina
assertEquals("GestioneCasino - Login",driver.getTitle());//controllo di essere arrivato alla pagina correttamente confrontando il titolo corretto con quel
```

< >

Un altro problema risolto è quello della creazione degli elementi, dove si provava a definire un WebElement per più situati in pagine diverse pur essendo uguali. Questo problema si è risolto semplicemente riassegnando l'elemento ogni volta, come mostrato nel codice precedente.

## Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

## Programma di massima per la prossima giornata di lavoro

Matan deve proseguire con la documentazione.

Matteo e Carlo devono risolvere il problema delle cartelle non apribili.

Thor proverà ad usare la versione più recente di JUnit per fare i prossimi test (JUnit 5 invece che JUnit 4).

# GESTIONE CASINÒ | Diario di lavoro - 29.03.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 29 marzo 2019

## Lavori svolti

Matan ha proseguito con la documentazione del progetto, continuando il capitolo 1.6, Pianificazione.

Thor ha installato insieme all'aiuto di Matan l'ultima versione di JUnit, scaricando tutte le jar di JUnit Jupiter dal sito:

<https://search.maven.org/search?q=g:org.junit.jupiter>

Successivamente gli ha Aggiunti alla Build Path del progetto di Eclipse, per poi aggiungere sempre dalla Build Path, JUnit 5.

Inoltre ha iniziato a documentarsi per fare in modo di utilizzare il WebDriver senza GUI, ed ha trovato principalmente 2 modi: JUnitHtmlDriver e PhantomJSDriver, di cui ha iniziato a implementare col primo test già funzionante, tuttavia senza successo.

Matteo e Carlo hanno continuato a lavorare su Jenkins risolvendo il problema delle cartelle non apribili che venivano create, per fare ciò le abbiamo semplicemente inserite nel file .gitignore creato appositamente.

GestioneCasino\*

test

githubscript.sh

.gitignore

Fatto ciò abbiamo scoperto che vi era un ulteriore problema di permessi, Jenkins non poteva escludere le cartelle e i files che vi erano nel file gitignore.

```
cp: cannot remove './GestioneCasinoProduction/GestioneCasino/.git/description': Permission denied
cp: cannot remove './GestioneCasinoProduction/GestioneCasino/.git/info/exclude': Permission denied
cp: cannot remove './GestioneCasinoProduction/GestioneCasino/.git/hooks/pre-push.sample': Permission denied
cp: cannot remove './GestioneCasinoProduction/GestioneCasino/.git/hooks/commit-msg.sample': Permission denied
cp: cannot remove './GestioneCasinoProduction/GestioneCasino/.git/hooks/fsmonitor-watchman.sample': Permission denied
cp: cannot remove './GestioneCasinoProduction/GestioneCasino/.git/hooks/prepare-commit-msg.sample': Permission denied
```

Risolto questo problema sono stati eseguiti dei test di push ed abbiamo potuto affermare che Jenkins funziona correttamente: ad ogni push esegue i test e, se risultano corretti, copia tutto nel repo di produzione e lo pusha.

Fatto ciò abbiamo provato a modificare la cartella dove apache tiene i files cambiandola con

```
/var/lib/jenkins/workspace/GestioneCasinoProduction
```

ma abbiamo avuto, anche qui, dei problemi di permessi.

Mattia ha creato la pagina dopo aver effettuato il login. La pagina conterrà la possibilità di modificare i propri dati (es: nome, cognome,...), inoltre si avrà la possibilità di modificare la password. Infine verranno mostrato un div in funzione dei diritti dell'utente, se è un admin avrà la possibilità di aggiungere o rimuovere utenti, sale, giochi e promozioni. Se invece è solamente un utente normale questo avrà la possibilità attraverso un pulsante di visualizzare tutte le promozioni del casinò. La pagina di mostra così:

# Benvenuto/Bentornato Nome



## Informazioni Base:

Nome: Gian

Cognome: Piero

Nascita: 10-12-2000

[Modifica Dati](#)

## Modifica Password:

Email: prova@prova.prova

[Modifica Password](#)

## Gestione Utenti:

Puoi gestire tutti gli utenti del sito, aggiungere o togliere qualsiasi utente:

[Modifica Utenti](#)

In seguito ha creato già la pagina per modificare le informazioni ed è molto simile alla pagina di registrazione solo con 4 input in meno: Sesso, Email, Passw ord e Ripeti Passw ord. Infine ha collegato la possibilità di modificare la passw ord alla pagina di modifica passw ord già creata nelle precedenti settimane. Il link è stato fatto così:

```

 <input type="button" class="form-control" value="Modifica Password" id="modify-password">

```

## Problemi riscontrati e soluzioni adottate

Matan è riuscito a "risolvere" il problema del capitolo 1.4 della documentazione, Analisi del dominio, scrivendo che l'applicazione è partita da zero.

Thor ha avuto qualche problema con le librerie, siccome richiamando sia quella di JUnitHtmDriver che quella di PhantomJSdriver(entrambi i metodi per usare il WebDriver senza GUI), usciva l'errore:  
cannot be resolved

Il problema riscontrato da Matteo e Carlo era che Jenkins non aveva i permessi per escludere le cartelle inserite nel .gitignore. Abbiamo inizialmente provato, nello script, a fare sì che Jenkins cambiasse l'utente utilizzato in quello di root e per farlo abbiamo utilizzato degli script trovati su internet ma nessuno era funzionante ed abbiamo quindi deciso di cambiare approccio. Come seconda prova abbiamo cambiato i permessi su tutti i files e questo ha risolto i nostri problemi.

Il secondo problema avuto da Carlo e Matteo è stato appunto quello di permessi sulla cartella dei files di apache. Per risolvere questo problema abbiamo seguito questa guida:

```
https://askubuntu.com/questions/413887/403-forbidden-after-changing-documentroot-directory-apache-2-4-6
```

che ci ha consentito di risolvere il problema dei permessi.

## Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

## **Programma di massima per la prossima giornata di lavoro**

---

Matan deve proseguire con la documentazione.

# GESTIONE CASINÒ | Diario di lavoro - 03.04.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 3 aprile 2019

## Lavori svolti

Oggi Matan è finalmente riuscito a far funzionare le asserzioni nei test di Selenium in modo che essi producano errori in caso di test non passati. Ha infatti dovuto creare, con l'aiuto di IntelliJ (usando la scorciatoia CTRL + SHIFT + T), una classe di test "SeleniumTestTest" che contiene il seguente metodo, che asserisce che due stringhe sono uguali:

```
class SeleniumTestTest {

 @Test
 void testTitle(String expected, String title) {

 assertEquals(expected, title);

 }
}
```

Ha poi dovuto aggiungere alla classe principale il seguente codice:

```
SeleniumTestTest st = new SeleniumTestTest();
st.testTitle("Casinò - Home", titleText);
```

Inoltre ha dovuto istanziare l'oggetto di tipo ChromeDriver utilizzato per navigare tra le pagine con un'istanza di ChromeOptions a cui è stato aggiunto il parametro "headless":

```
ChromeOptions options = new ChromeOptions();
options.addArguments("headless");
WebDriver driver = new ChromeDriver(options);
```

Oggi Matteo ha controllato che Jenkins funzionasse correttamente controllando che tutti i push della giornata generassero il risultato voluto sulla repo GestioneCasinoProduction. Fatto ciò ha provato a riprendere il lavoro svolto da Thor ma non è stato possibile dato che le sue prove non si trovavano sul git. Ha dunque chiesto a Matan il suo codice per capirne il funzionamento ed ha iniziato a cercare come eseguirli senza un'interfaccia grafica. Per fare ciò ha trovato il seguente sito: <https://www.blazemeter.com/blog/headless-execution-selenium-tests-jenkins> che spiega come eseguire i test di Selenium tramite quelli di Jenkins.

Mattia e Carlo oggi si sono dedicati alla pagina del dopo login. Più precisamente Mattia si è occupato di creare la pagina front-end per la gestione delle sale, giochi, utenti e promozioni. Questa pagina è composta da un input di tipo testo per il titolo, un input di tipo file per aggiungere un'immagine e una textarea per la descrizione. Per l'input di tipo file abbiamo cercato un template che permetesse di aggiungere l'immagine ,oltre che al pulsante, attraverso un drag and drop:

<https://bootsnipp.com/snippets/DOxY4>. Il codice aggiunto per questo input è il seguente:

HTML:

```
<div class="row">
 <div class="col-md-6">
 <div class="form-group files color">
 <input type="file" class="form-control" accept="image/*">
 </div>
 </div>
</div>
```

CSS:

```
<div class="row">
.files input {
 outline: 2px dashed #92b0b3;
 outline-offset: -10px;
 -webkit-transition: outline-offset .15s ease-in-out, background-color .15s linear;
 transition: outline-offset .15s ease-in-out, background-color .15s linear;
 padding: 120px 0px 85px 35%;
 text-align: center !important;
 margin: 0;
 width: 100% !important;
}
.files input:focus{
 outline: 2px dashed #92b0b3;
```

```

outline-offset: -10px;
-webkit-transition: outline-offset .15s ease-in-out, background-color .15s linear;
transition: outline-offset .15s ease-in-out, background-color .15s linear; border:1px solid #92b0b3;
}
.files{ position:relative}
.files:after { pointer-events: none;
position: absolute;
top: 60px;
left: 0;
width: 50px;
right: 0;
height: 56px;
content: "";
background-image: url(https://image.flaticon.com/icons/png/128/109/109612.png);
display: block;
margin: 0 auto;
background-size: 100%;
background-repeat: no-repeat;
}
.color input{ background-color:#f1f1f1;}
.files:before {
position: absolute;
bottom: 10px;
left: 0; pointer-events: none;
width: 100%;
right: 0;
height: 57px;
display: block;
margin: 0 auto;
color: #2ea591;
font-weight: 600;
text-transform: capitalize;
text-align: center;
}

```

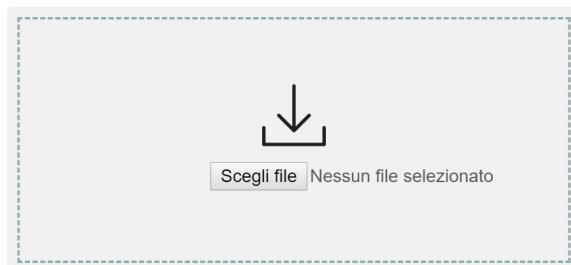
Il risultato finale della pagina è il seguente:

## Inserire dati

Titolo:

Titolo

Immagine:



Descrizione:

Inserisci una descrizione...

**AGGIUNGI**

Carlo invece si è dedicato alla parte back-end della pagina di aggiunta delle sale, giochi, promozioni e utenti. Innanzitutto adatta il titolo della pagina al contesto, nel senso che quando per dire l'admin clicca sul pulsante per modificare le sale il tipo deve essere "Modifica sale". Questa cosa è gestita grazie al metodo GET. Poi ha creato un form che si adatta in modo dinamico per aggiungere le varie cose (sale, giochi, promozioni e utenti), questo è fatto così:

```

$get = $_GET['type'];
try{
 $result = $db->executeQuery("select * from $get");
}

```

```

$n = 0;
foreach ($result[0] as $key => $value) {
 if($n%2==0){
 echo "<div class='col-md-12 col-sm-12'>
 ".strtoupper($key{0}).substr($key,1,strlen($key))."
 <input name='".$key' type='text' class='form-control' id='title' placeholder='".strtoupper($key{0}).substr($key,1,strlen($key))."'>
 </div>";
 }
 $n++;
}
}catch(InvalidArgumentException $iae){}

```

Infine assieme alla cosa soprastante ha gestito l'inserimento in modo dinamico dei campi:

```

if ($_SERVER["REQUEST_METHOD"] == "POST") {
 $get = $_GET["type"];
 $query = "Insert into $get(";
 $n = 0;
 foreach ($_POST as $key => $value) {
 if($n+1 == sizeof($_POST))
 $query .= $key;
 else
 $query .= $key . ",";
 $n++;
 }
 $query .= ") values(";
 $n = 0;
 foreach ($_POST as $key => $value) {
 if($n+1 == sizeof($_POST))
 $query .= "'$value'";
 else
 $query .= "'$value'" . ",";
 $n++;
 }
 $query .= ")";
 //echo $query;
 $db->executeQuery($query);
 header("Refresh:0");
}

```

Thor le prime 2 ore ha fatto il test del modulo 133, successivamente nel tempo che gli rimaneva a cercato dei modi per far andare selenium nel server linux (senza GUI), su stackoverflow:

<https://stackoverflow.com/questions/34342632/why-selenium-work-on-ubuntu-with-no-gui>

Dove consigliano di usare "xvfb" dopodiché ha letto anche la guida trovata da Matteo per far funzionare i test di Selenium con Jenkins.

Inoltre ha provato continuato a creare il codice Java per la registrazione degli utenti (con file csv, iniziato la scorsa lezione).

## Problemi riscontrati e soluzioni adottate

### Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

### Programma di massima per la prossima giornata di lavoro

Matan deve cercare di far funzionare il test di Selenium completato oggi con Firefox e informarsi su come fare a eseguire il test senza un'interfaccia grafica.

Matteo deve provare ad eseguire dei test senza GUI.

Thor deve trovare un modo per far andare i test di selenium sul server e continuare a cercare codice di test(nel dettaglio quello della registrazione, se finisce ne inizierà un altro).

# GESTIONE CASINÒ | Diario di lavoro - 05.04.2019

Matan Davidi, Thor Düblin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 5 aprile 2019

## Lavori svolti

Matan ha proseguito con la documentazione del progetto, aggiungendo i capitoli 1.6.3, Progettazione, 1.6.4, Implementazione, 1.6.6, Documentazione di progetto e 6.2, Considerazioni personali.

Thor e Matteo oggi hanno cercato un modo per far andare i test di selenium sul server, creando una virtuale di prova del server e seguendo i passaggi mostrati in questa guida che sono andati a buon fine:

<https://tecadmin.net/setup-selenium-chromedriver-on-ubuntu/>

Tuttavia per installare chrome si è usata un'altra guida siccome il capitolo riguardante quell'operazione non andava a buon fine:

<https://linuxize.com/post/how-to-install-google-chrome-web-browser-on-ubuntu-18-04/>

Successivamente si ha provato a utilizzare i test di prova tuttavia senza successo, siccome la libreria installata non era nella stessa posizione del file Java.

Carlo ha proseguito con la stesura del codice backend. Durante lo stand-up ci siamo chiariti su un paio di cose che precedentemente erano sconosciute:

- la prima era che pagina mostrare una volta eseguito il login -> la soluzione è stata quella di mostrare la pagina dove vengono mostrati le possibili interazioni che l'utente può fare (modificare la password, le informazioni e visualizzare le promozioni).
- la seconda era la gestione dell'accesso da parte di un utente, ovvero -> se un utente vuole essere ricordato allora le sue info vengono salvate un cookie (email) con la vita di 7 giorni. Altrimenti vengono usate le sessioni con lo stesso principio.

Il codice che ho sviluppato oggi non è nulla di che. Mi sono occupato di gestire la pagina di profilo di un utente.

```
<?php
require_once "php/loader.php";
session_start();
if(empty($_SESSION["username"])){
 setcookie("error","Pagina non trovata :(", time() + 1000,"/");
 header("Location: error.html");
 exit();
}

$queryRepose = $db->executeQueryWithoutFetch("select * from user where email = '". $_SESSION['username']."'")->fetch();
?>
```

```
<p>Nome: <?php echo $queryRepose["name"]?></p>
<p>Cognome: <?php echo $queryRepose["surname"]?></p>
<p>Nascita: <?php echo $queryRepose["birthday"]?></p>
<input type="button" class="form-control" value="Modifica Dati">
```

Oggi Mattia ha creato la pagina delle sale, dei giochi e delle promozioni. Tutte e tre hanno lo stesso formato, perché vengono ogni volta che si vuole aggiungere una sala, un gioco o una promozione si deve adattare dinamicamente grazie al pannello che dispone l'admin per le aggiunte di quest'ultime. Ogni volta che l'admin aggiunge per esempio una sala, nella pagina viene aggiunto un div che contiene il titolo (della sala), un'immagine e una breve descrizione. Questo div viene realizzato così:

```
<div class="blog-post-thumb">
 <div class="blog-post-image" id="[" . num . "]>

 </div>
 <div class="blog-post-title">
 <h3>[Titolo]</h3>
 </div>
 <div class="blog-post-des">
 <p>[Descrizione]</p>
 </div>
</div>
```

## Problemi riscontrati e soluzioni adottate

Oggi Matan e Mattia hanno avuto un problema di comprensione dei requisiti. Infatti Mattia ritiene che le promozioni devono essere mostrate a tutti gli utenti con titolo, immagine e descrizione, mentre Matan pensa che esse debbano essere composte unicamente dal messaggio e mostrate solo ad alcuni utenti. È stata mandata un'email al cliente per chiarimenti a riguardo e abbiamo intenzione di parlarne al prossimo stand-up.

## **Punto della situazione rispetto alla pianificazione**

---

Rispetto alla pianificazione siamo in orario.

## **Programma di massima per la prossima giornata di lavoro**

---

Per la prossima lezione l'obiettivo di Thor e Forni sarà far funzionare i test sul server di prova e successivamente sul server originale.

# GESTIONE CASINÒ | Diario di lavoro - 10.04.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 10 aprile 2019

## Lavori svolti

Grazie alla dritta data dal cliente, Matan ha potuto modificare il database, aggiungendo la colonna "type" alla tabella "user" che contiene il tipo di utente in modo da potergli mostrare o meno una determinata promozione. Inoltre ha modificato la tabella "media" in modo che non contenga riferimenti ad altre tabelle, e sono state aggiunte tre tabelle ponte al loro posto, una tra "game" e "media", "game\_media", una tra "promotion" e "media", "promotion\_media", e una tra "room" e "media", "room\_media". Infine ha aggiunto una tabella che contiene i valori predefiniti per i tipi di utente chiamata "user\_type" e ha rinominato "type" in "media\_type".

Adesso lo schema logico del database è il seguente:

- game(room, name, description)
- game\_media(game\_name, game\_room, media\_url)
- gender(name)
- media(url, type)
- media\_type(name)
- promotion(id, name, description)
- promotion(promotion\_id, media\_url)
- promotion\_user(user\_type, promotion\_id)
- room(location, description)
- room\_media(room\_location, media\_url)
- user(name, surname, street, house\_number, zip\_code, city, email, phone\_number, gender, password, type, admin)
- user\_type(name)

Oggi Thor si è concentrato inizialmente sul risolvere il problema che ha riscontrato l'ultima volta riguardante i test sul server, inizialmente ha lavorato su un server con GUI, per poi passare a installare un server senza GUI così da controllare se i test funzionino senza, un altro problema riscontrato sono le license mancanti siccome su VMWare Workstation sono scadute.

Oggi Carlo si è occupato di continuare con lo sviluppo backend, ovvero ho messo a posto i test su database per via dei cambiamenti visti. Inoltre ho aggiunta la possibilità di eliminare o modificare un utente. Il codice che fa richieste al database però è ancora stato fatto. Codice di per sé nuovo non c'è stato perché ho solo modificato poche righe di codice nella classe database.

Oggi Matteo ha ripreso il lavoro fatto in passato da Thor. Dopo aver installato Maven, IntelliJ ed aver scaricato il chromedriver ha copiato un vecchio progetto di Matan e vi ha inserito il codice di Thor per verificarne il funzionamento. Dopo aver corretto un paio di percorsi sbagliati i test funzionavano e quindi ha proseguito creando un metodo per ogni pagina così da rendere il codice più pulito.

```
void home() {
 WebElement home = null;
 home = driver.findElement(By.linkText("Home"));
 home.click();
 wait();
 System.out.println(driver.getTitle());
 assertEquals(expected: "Casinò - Home", driver.getTitle());
}
```

Questo è stato fatto perché prima vi erano molte ripetizioni nel codice ed esso diventava inutilmente lungo.

Fatto ciò ha aggiunto i test dei link che mancavano.

Oggi Mattia ha fatto un merge tra la propria versione con quella di Carlo. Non ha avuto particolari problemi se non con un paio di file i quali erano stati modificati da entrambi, quindi ci siamo dovuti accordare. Inoltre a messo a posto delle piccolezze tra le pagine html, come nomi, titoli di pagina e link tra le varie pagine.

## Problemi riscontrati e soluzioni adottate

Carlo: Non riuscivo a capire perché i test in locale sul database non andassero, alla fine ho scoperto che il servizio MySQL era buggato e con un riavvio ho risolto tutto

## Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

## **Programma di massima per la prossima giornata di lavoro**

---

Matan deve modificare e/o aggiungere le foreign key al database che ha modificato oggi e portarlo sul server di produzione DigitalOcean.

Thor la prossima lezione testerà i test di selenium su un server senza interfaccia grafica, e successivamente continuerà a programmare il test di Registrazione.

Carlo la prossima volta si occuperà di fare le richieste al database per eliminare o modificare gli utenti già esistenti

# GESTIONE CASINÒ | Diario di lavoro - 12.04.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 12 aprile 2019

## Lavori svolti

Oggi Matan è riuscito ad aggiungere le foreign key al nuovo database, cominciato la lezione precedente, e l'ha esportato in [questo file](#) con il seguente comando:

```
mysqldump --add-drop-table -u user -ppassword cashyland > cashyland_db_2.sql
```

In seguito ha sostituito il database precedente con quello nuovo importandolo con:

```
mysql -u user -ppassword cashyland < cashyland_db_2.sql
```

Matteo oggi ha continuato i test su Selenium completando, senza problemi, quello di registrazione e quello di login. Per capire come scrivere negli input ha utilizzato il seguente sito: <https://www.guru99.com/accessing-forms-in-w-ebddriver.html>.

Per entrambe le due classi di test la struttura utilizzata è stata quella della classe di test dei link quindi con un metodo per ogni pagina da utilizzare ed uno per ogni input in cui scrivere. I metodi degli input sono tutti molto simili.

```
void insertName(String name) {
 WebElement nome = driver.findElement(By.id("firstname"));
 nome.sendKeys(...keysToSend: name);
}

void insertSurname(String surname) {
 WebElement cognome = driver.findElement(By.id("surname"));
 cognome.sendKeys(...keysToSend: surname);
}
```

Per il test di login si utilizza sempre lo stesso utente creato appositamente.

Thor intanto oggi si è occupato di continuare la creazione di un server senza GUI con selenium dove poter testare definitivamente i test prima di installare il tutto nel server originale, riscontrando problemi nella configurazione del proxy, superati utilizzando una connessione non scolastica.

Carlo oggi ha gestito l'errore legato al fatto che un utente non può registrarsi un email già esistente. Inoltre ho riscontrato molti errori legati ai test fatti in phpUnit. Con la modifica del database ho dovuto modificare i test ma senza successo.

Mattia oggi ha messo a posto altri title e link della pagina. In seguito dopo vari test di Matteo si è scoperto che i messaggi di errore della pagina di registrazione non sono molto esplicativi, cioè non si capisce molto bene quale sia l'errore. Allora Mattia ha pensato di fare un tooltip che quando si scrive esce e dice come bisogna compilare il campo. Dopo vari tentativi di aggiunta non ci è riuscito, ma solo in seguito ha scoperto che si può utilizzare la libreria notify.js. Quindi ha cominciato a farlo così:

```
onfocus="tooltip(this,'casa')"
```

e nel file javascript

```
//Metodo che simula un tooltip
function tooltip(input, testo){
 $(input).notify(testo, "info");
}
```

## Problemi riscontrati e soluzioni adottate

Matan ha avuto problemi a importare il dump del database nel server di produzione, forse a causa delle sottili differenze tra MySQL e MariaDB. La soluzione adottata è stata quella di ritoccare a mano il file esportato in modo che sia accettabile dal DBMS di produzione.

Oggi il servizio di hotmail ci ha bloccati ancora quindi il servizio mail non andava più. Ho cercato di metterlo a posto ma non andava comunque. Allora abbiamo optato di cambiare servizio. Ora usiamo gmail.

## **Punto della situazione rispetto alla pianificazione**

---

Rispetto alla pianificazione siamo in orario.

## **Programma di massima per la prossima giornata di lavoro**

---

Importare il database aggiornato sul server di produzione.

# GESTIONE CASINÒ | Diario di lavoro - 17.04.2019

Matan Davidi, Thor Düblin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 17 aprile 2019

## Lavori svolti

Matan è riuscito a importare correttamente il database aggiornato sul server di produzione modificando il file dump estratto dalla sua versione locale fino a farlo diventare il contenuto di [questo file](#). Di seguito trovate un breve riassunto delle modifiche:

- Aggiunte tabelle ponte "media" - "game", "media" - "room" e "media" - "promotion".
- Aggiunti attributi "title" e "description" dove necessario in modo che compaiano nelle tabelle "game", "room" e "promotion".
- Aggiunta la colonna "type" di tipo VARCHAR(45) alla tabella "user". Questa colonna ha un vincolo di integrità che punta alla tabella "user\_type", che contiene dei valori predefiniti.
- Cambiato il nome della tabella "type" in "media\_type" per non confonderla con "user\_type".
- Modificata la tabella promotion\_user in modo che si possa mostrare una promozione a un certo tipo di utenti invece che a un utente singolo (questo è stato fatto modificando la colonna "user\_name" in "user\_type" e impostando la chiave esterna in modo che si riferisca alla colonna "type" della tabella "user").

Inoltre ha aggiunto i valori predefiniti "Maschio" e "Femmina" alla tabella "gender" e i valori "occasionale" e "settimanale" nella tabella "user\_type".

Thor oggi ha completato l'installazione di selenium in modo che possano essere eseguiti i test nel server senza interfaccia grafica, installando quindi diversi pacchetti tra cui:

- xvfb,
- libxi6,
- libgconf-2-4,
- google-chrome-stable,
- chromedriver\_linux64.zip,
- openjdk-8-jre-headless,
- default-jdk e infine
- selenium-server-standalone-3.13.0.jar e
- testng-6.8.7.jar.zip

Per eseguire, i test le librerie selenium-server-standalone-3.13.0.jar e testng-6.8.7.jar dovranno essere presenti nella stessa cartella dei test, e prima di compilare la classe per eseguire i java dovrà essere scritto il seguente codice da linea di comando:

```
export CLASSPATH=".:selenium-server-standalone-3.13.0.jar:testng-6.8.7.jar"
```

Per poi compilare il programma di test con javac ed eseguirlo con java.

Per completare l'installazione correttamente si sono seguite le guide già citate in precedenza:

- <https://tecadmin.net/setup-selenium-chromedriver-on-ubuntu/>
- <https://gist.github.com/ziadoz/3e8ab7e944d02fe872c3454d17af31a5>

Matteo oggi ha continuato e terminato i test con selenium. Le modifiche apportate sono che ora il test di registrazione controlla almeno tre casi errati per ogni input dopodiché, se tutti giustamente falliscono, fa una prova con dei dati validi.

Nel test di login sono è stato aggiunto il metodo che esegue il click del bottone ed in entrambe le classi sono state fatte alcune piccole modifiche dovute a dei cambiamenti della pagina html.

Il test della registrazione si presenta quindi così:

```

/*Test name*/
testTextInputs(text: "aa", input: "name");
testTextInputs(text: "21", input: "name");
testTextInputs(text: "Selenium", input: "name");
assertEquals(expected: "CashtLand - Registrazione", driver.getTitle());
System.out.println("NAME OK");

```

Per ogni input si esegue un test come mostrato, i metodi di test riempiono tutti gli input con valori validi tranne quello da testare così da verificare se la validazione di ogni input singolarmente funziona.

Carlo oggi ha messo a posto il test legati ai controlli del database. Durante lo stand-up abbiamo pensato che su il repository production ci deve essere solo il sito, quindi ho modificato lo script per il push sulla repo.

Mattia oggi ha messo a posto le notifiche mostrate a schermo quando l'utente inserisce i propri dati nella pagina di registrazione. La lezione scorsa non era riuscito a modificare in modo dinamico il testo delle notifiche e quest'oggi a messo a posto nel modo seguente:

```

<input
 name="surname"
 type="text"
 class="form-control"
 id="surname" placeholder="Cognome"
 onkeydown="normal(this)"
 onfocus="tooltip(this,' Max 50 caratteri, solo lettere e caratteri da scrittura')"
>

```

In seguito nella funzione tooltip viene mostrata la notifica all'input corrispondente nella posizione in basso a destra così:

```

function tooltip(input, testo){
 $(input).notify(
 testo,
 {
 elementPosition:"bottom right",
 className: 'info'
 }
);
}

```

## Problemi riscontrati e soluzioni adottate

Matan ha avuto problemi a installare Selenium sul server, prima a causa di un problema di memoria RAM, risolto chiudendo il processo di jenkins, veniva mostrato il seguente errore:

```

dpkg: unrecoverable fatal error, aborting:
 unknown group 'smmsp' in statoverride file
E: Sub-process /usr/bin/dpkg returned an error code (2)

```

Questo problema è stato risolto rimuovendo i riferimenti al gruppo creato dall'installazione del mail server eseguita in passato. Matan ha infatti eseguito i seguenti comandi:

```

dpkg-statoverride --remove /usr/lib/sm.bin/sendmail
dpkg-statoverride --remove /usr/lib/sm.bin/mailstats

```

Matteo ha riscontrato come problema che Selenium negli input non sovrascrive ma aggiunge solo test, ha dovuto quindi creare un metodo che gli svuotasse ogni volta.

```
void clearInputs(){
 nome.clear();
 cognome.clear();
 dataNascita.clear();
 via.clear();
 noCivico.clear();
 cap.clear();
 citta.clear();
 telefono.clear();
 indirizzoEmail.clear();
 pass.clear();
 repass.clear();
}
```

## Punto della situazione rispetto alla pianificazione

---

Rispetto alla pianificazione siamo in orario.

## Programma di massima per la prossima giornata di lavoro

---

Controllare che i test scritti finora funzionino anche senza interfaccia grafica.

# GESTIONE CASINÒ | Diario di lavoro - 03.05.2019

---

Matan Davidi, Thor Düblin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 3 maggio 2019

## Lavori svolti

---

Matan oggi ha continuato il capitolo 1.6, Pianificazione, aggiungendo la parte di progettazione, implementazione e di protocolli di test. Inoltre ha rimosso l'ultimo sottocapitolo della parte di progettazione, 2.3 - Design delle interfacce, ritenuto inutile in quanto non v'era niente da documentare.

Thor oggi ha installato le librerie junit necessarie per far andare i test, cioè:

-apiguardian-api-1.0.0.jar  
-junit-jupiter-api-5.4.2.jar  
-junit-platform-commons-1.4.2.jar  
-opentest4j-1.1.1.jar"

Che verranno aggiunti al export CLASSPATH, ogni prima che si vorranno eseguire i test, nel seguente modo:

```
export CLASSPATH=.:selenium-server-standalone-3.13.0.jar:testng-6.8.7.jar:apiguardian-api-1.0.0.jar:junit-jupiter-api-5.4.2.jar:junit-platform-commons-1.4.2.jar:opentest4j-1.1.1.jar"
```

Dopo aver eseguito questo codice e eseguendo il file di test java per qualche ragione Chrome crashava, credendo che questo errore non sia causato test, ci si è documentati nel poco tempo che rimaneva, senza però trovare una soluzione.

Mattia e Carlo oggi hanno iniziato a fare la parte di implementazione della documentazione in quanto, discutendo con il gruppo, hanno pensato che è meglio prima concludere (o quasi) la documentazione invece che concludere il progetto e magari non riuscire a documentarlo. Queste parti di documentazioni sono state scritte in locale per non creare conflitti nella file principale.

Matteo oggi ha completato i test di Selenium aggiungendo il controllo dei permessi dell'utente. Ha inoltre cambiato i percorsi del chromedriver in tutti i programmi e risolto alcuni bug.

## Problemi riscontrati e soluzioni adottate

---

Successivamente l'immagine dell'errore riscontrato riguardo il malfunzionamento di Chrome:

]

Il problema di Matteo era che su linux il metodo .clear() non funziona con gli input di tipo data. Ha quindi dovuto cambiarlo con il seguente codice.

```
dataNascita.sendKeys(Keys.CLEAR);
```

## Punto della situazione rispetto alla pianificazione

---

Rispetto alla pianificazione siamo in orario.

## Programma di massima per la prossima giornata di lavoro

---

Continuare con la documentazione.

# GESTIONE CASINÒ | Diario di lavoro - 08.05.2019

---

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 8 maggio 2019

## Lavori svolti

---

Oggi Matan ha proseguito con la documentazione, modificando la data di consegna, come riferiti dal cliente, al 22 maggio invece del 17 maggio. Inoltre ha rimosso il capitolo 2.1, Design dell'architettura del sistema, ritenuto inutile in quanto non vi sarebbe nulla da documentare. Poi ha aggiunto i diagrammi UML e le relative immagini di tutte le classi scritte in PHP all'interno del capitolo 2.2, Design procedurale. Infine ha aggiunto il capitolo 6, Conclusioni.

Oggi Thor per le prime 2 ore si è soffermato sul problema riguardo ai crash di Chrome quando si avviavano i test di Selenium, provando ad aggiungere e modificare ChromeOptions del WebDriver, tuttavia senza trovare ancora soluzione a questo problema, successivamente si è occupato della documentazione.

Oggi Matteo ha completato tutti i test di Selenium aggiungendo anche le chrome options così da poter eseguire il tutto senza un'interfaccia grafica.

```
System.setProperty("webdriver.chrome.driver", "/usr/bin/chromedriver");
ChromeOptions options = new ChromeOptions();
options.addArguments("--headless");
options.addArguments("--no-sandbox");
options.addArguments("--disable-dev-shm-usage");

driver = new ChromeDriver(options);
```

Negli stessi files ha rimosso tutto il codice inutile ed ha modificato un paio di metodi per ottimizzare il codice. Fatto ciò si è dedicato alla documentazione iniziando a scrivere la parte di implementazione riguardante Jenkins ed i test di Selenium, il testo è momentaneamente in un file in locale in attesa di aggiungere il testo alla documentazione ufficiale. Completata la propria parte di documentazione ha inoltre provato a risolvere il problema riscontrato da Thor senza però riuscire a trovare una soluzione.

Mattia e Carlo oggi hanno continuato a fare la parte di implementazione della documentazione. Queste parti di documentazioni sono state scritte in locale per non creare conflitti nella file principale.

## Problemi riscontrati e soluzioni adottate

---

Il bottone "Scopri di più" non porta in basso alla pagina se si sta visitando il sito con Google Chrome.

## Punto della situazione rispetto alla pianificazione

---

Rispetto alla pianificazione siamo in orario.

## Programma di massima per la prossima giornata di lavoro

---

Continuare con la documentazione.

Matteo ha intenzione di far funzionare una volta per tutte Selenium sul server di produzione ed aggiungere i test di selenium al lavoro di Jenkins.

# GESTIONE CASINÒ | Diario di lavoro - 10.05.2019

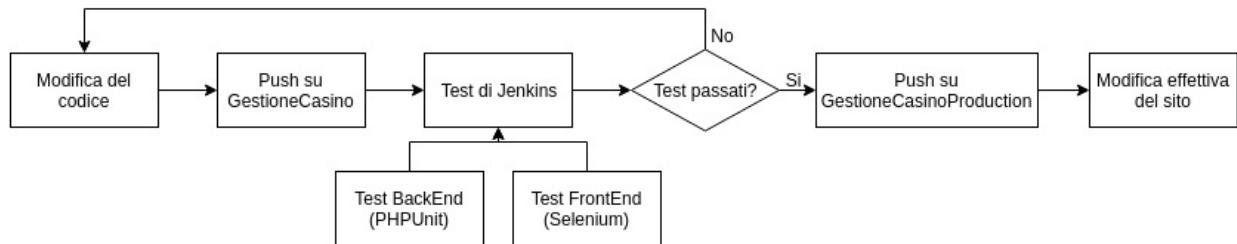
Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 10 maggio 2019

## Lavori svolti

Durante la giornata di lavoro di oggi, Matan ha continuato a documentare il progetto, rimuovendo qualche testo segnaposto lasciato dal template, aggiungendo nuovamente il capitolo 2.1, Design dell'architettura del sistema, rimosso l'ultima lezione, contenente il diagramma di flusso dell'applicazione dalla modifica del codice alla messa in produzione, attraverso Jenkins, PHPUnit e Selenium realizzato da Matteo, creando e inserendo i diagrammi UML delle classi Database e MailSender e aggiungendo una descrizione a tutto il sotto-capitolo di PHP all'interno del capitolo 2.3, Design procedurale.

Oggi Matteo ha creato un diagramma di flusso che possa spiegare e rappresentare al meglio il flusso di lavoro del sito e dei vari software che ci stanno dietro come Jenkins e Selenium.



Dopodiché si è occupato di aiutare Thor nella risoluzione dei problemi di Selenium senza ottenere però grandi risultati.

Oggi Thor si è occupato di risolvere i problemi legati ai test di selenium, risolvendo l'errore riscontrato sui crash di chrome utilizzando firefox e geckodriver, tuttavia si è riscontrato un problema legato alla connessione e all'impossibilità utilizzare porte "DISPLAY".

Carlo oggi ha concluso la parte di gestione delle sale, dei giochi e delle promozioni. Per far il tutto ha usato la mia pagina che va ad interpellare il database in modo dinamico, già mostrato e spiegato qualche diario fa. L'aggiunta è stata una parte dinamica che va in base al parametro passato come get, il seguente codice illustra il tutto.

```
if($_GET["type"] == "room"){
 echo'
 <input type="button" class="form-control" value="Aggiungi un immagine alla sala!">
 ';
} else if($_GET["type"] == "game"){
 echo'
 <input type="button" class="form-control" value="Aggiungi un immagine al gioco!">
 ';
} else if($_GET["type"] == "promotion"){
 echo'
 <input type="button" class="form-control" value="Aggiungi un immagine alla promozione!">
 ';
}
```

Il seguente codice non fa altro che rimandare l'utente sulla stessa pagina con però un tipo di parametro diverso. L'ultima cosa che ho fatto è stata la possibilità di aggiungere immagini.

```
require_once "php/loader.php";
$target_dir = "mediaFiles/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
$message = "";
if(isset($_POST["ok"])) {
 if (file_exists($target_file)) {
 $message .= "Sorry, file already exists. ";
 $uploadOk = 0;
 }
 if ($uploadOk == 0) {
 $message .= "Sorry, your file was not uploaded. ";
 } else {
 if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
 $db->executeQuery("insert into media values('".$target_file."','".$imageFileType."')");
 $message.= "The file ". basename($_FILES["fileToUpload"]["name"]). " has been uploaded. ";
 } else {
 $message.="Sorry, there was an error uploading your file. ";
 }
 }
}
```

```
 }
 echo "<script>alert('$message');</script>";
}
```

Il seguente codice aggiunge un file sul server.

Mattia oggi ha continuato a fare la parte di implementazione della documentazione. Più precisamente le parti grafiche fatte con HTML e CSS Queste parti di documentazioni sono state scritte in locale per non creare conflitti nella file principale.

## Problemi riscontrati e soluzioni adottate

Thor ha risolto il problema legato ai crash di chrome utilizzando firefox e geckodriver al posto di chrome e rispettivamente chromedriver.  
Successivamente il codice che risolve il problema dei crash di chrome:

```
final File firefoxPath = new File(System.getProperty("limportal.deploy.firefox.path", "/usr/bin/firefox"));

String Xport = System.getProperty("limportal.xvfb.id", ":1");

driver = new FirefoxDriver(new GeckoDriverService.Builder()
 .usingDriverExecutable(new File("/usr/bin/geckodriver"))
 .usingFirefoxBinary(new FirefoxBinary(firefoxPath))
 .withEnvironment(ImmutableMap.of("DISPLAY", Xport)).build());
```

Tuttavia in questo modo si sono verificati altri problemi legati alla connessione e all'impossibilità di utilizzare i DISPLAY nel server.

## Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

## Programma di massima per la prossima giornata di lavoro

Rinominare le classi e i progetti di Selenium in modo più semantico e farli funzionare.  
Continuare con la documentazione.

# GESTIONE CASINÒ | Diario di lavoro - 15.05.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 15 maggio 2019

## Lavori svolti

Oggi per la maggior parte della durata della giornata Matan ha cercato di sistemare i problemi causati da un errore che ha fatto Thor tra la lezione scorsa e questa, che ha causato alcuni ripristini di uno o più file a uno stato precedente.

Oggi Thor ha provato a risolvere i problemi legati ai test di Selenium che riguardavano, inizialmente gli errori erano legati alla connessione del server e al display della porta, questo si crede accadeva perché non si attivava la xvfb col seguente comando:

Xvfb :1 -screen 0 1024x768x24 &

successivamente si è riscontrato l'errore seguente:

INFO: HTTP Status: '404' -> incorrect JSON status mapping for 'unknow n error'

Pensando che si trattasse di un errore di versioni come trovato in rete, si ha provato ad aggiornare firefox, geckodriver e a startare i test con un non-root user. Questo non ha comunque risolto i problemi.

Oggi Matteo ha creato un test di Selenium che si è reso conto che mancava, esso testa la modifica dei dati di un utente da parte dello stesso. Esso è molto simile al test di login (dato che all'inizio l'utente deve loggarsi) e a quello di registrazione con la modifica dei dati nel form. Fatto ciò ha aggiunto al test di login i controlli che ci siano tutti i buttoni degli admin, che consentono di aggiungere utenti, sale, giochi, promozioni e immagini, e che essi portano alla pagina giusta. Facendo ciò ha dovuto cercare come si facesse a ritornare alla pagina precedente con Selenium, esso si fa nel modo seguente:

```
driver.navigate().back();
```

Fatto ciò ha scoperto che Jenkins non funzionava più correttamente, leggendo l'errore ha capito che il problema stava nella versione di Java appena installata da Thor. Jenkins infatti non funziona con versioni successive alla 8. Per risolvere è bastato quindi disinstallare l'ultima versione. Dopodiché Jenkins non funzionava ancora ed è servito un riavvio del server per riportare la situazione alla normalità.

Oggi Carlo ha modificato il metodo printTable nella classe database

```
$tableName = explode(" ",$selectQuery);
for($i = 1; $i < sizeof($tableName);$i++){
 if($tableName[$i-1] == "from"){
 $tableName = $tableName[$i];
 break;
 }
}
$primaryKey = $this->db->query("SHOW KEYS FROM ".$tableName." WHERE Key_name = 'PRIMARY'")->fetchAll()[0]["Column_name"];
```

Il seguente codice serve per identificare una primary key all'interno di un database usata poi nella stampa per mostrare all'utente quale colonna è effettivamente la primary key.

Inoltre ho finalmente concluso la possibilità di modificare e eliminare dei dati dal database lato amministrativo. Questo sistema l'ho pensato completamente dinamico almeno da farlo adattare a qualsiasi tabella necessaria. Il codice per l'eliminazione dei dati dal database è il seguente:

```
require_once "../loader.php";
if(isset($_GET["table"]) && isset($_GET["value"]) && isset($_GET["key"])){
 $db->executeQuery("Delete from ". $_GET["table"]." where ".$_GET["key"]." = '".$_GET["value"].".'");
 header('Location: ' . $_SERVER['HTTP_REFERER']);
} else{
 header('Location: ' . $_SERVER['HTTP_REFERER']);
}
```

Il seguente codice tramite parametri get riceve la tabella dove eliminare un dato, il nome della primary key e il suo valore.

```
require_once "../loader.php";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
 if(isset($_GET["table"])){
 $get = $_GET["table"];

 $n = 0;
 $query = "update ".$get." set ";
 foreach ($_POST as $key => $value) {
 if($n>1){
 if($n+1 == sizeof($_POST))
 $query .= $key." = '". $value. "'";
 else
 $query .= $key." = '". $value. "', ";
 }
 }
 }
}
```

```

 }
 $n++;
 }
 $n =0;
$query .= " WHERE ".$_POST["primaryKey"]." = '". $_POST["keyValue"]. "'";
//echo $query;
$db->executeQuery($query);
//$/db->executeQuery($query);
header("Location: ../../addThings.php?type=$get");
}
}

```

Il seguente codice tramite parametri get riceve la tabella dove eliminare un dato, il nome della primary key e il suo valore. Si occupa di andare a creare una query dinamica in base alle colonne del database e le modifica.

Mattia ha messo a posto molti piccoli difetti della nostra applicazione. Inanzitutto ha aggiustato i vari link da .html a .php. In seguito ha reso responsive la tabella delle modifiche sale/giochi/ffromozioni e utenti con bootstrap, nel seguente modo:

```

<div class="table-responsive text-nowrap">
 <!--Table-->
 <table class="table table-striped">

 </table>
</div>

```

Poi ha aggiunto vari id a input per eseguire vari test con Selenium. Infine ha continuato la documentazione per quando riguarda la descrizione delle pagine.

## Problemi riscontrati e soluzioni adottate

I file che hanno causato dei problemi in seguito all'errore commesso da Thor sono i seguenti:

- **code/java/selenium/chrome/navigation/SeleniumTest.idea/workspace.xml**, che causava al progetto SeleniumTest di non riuscire più a leggere le proprie impostazioni. Questo è stato sistemato cancellando il file, aprendo il progetto, selezionando File, poi Invalidate Caches / Restart, riavviando il software e aggiungendo nuovamente la configurazione d'avvio.
- **documentation/2019.05.08\_i3\_davidi\_dueblin\_forni\_pezzotti\_toscanelli\_conclusioni.docx**, con il relativo capitolo della documentazione. Entrambi sono stati eliminati e Matan ha quindi dovuto ripristinarli da un commit precedente.

## Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione siamo in orario.

## Programma di massima per la prossima giornata di lavoro

Rinominare le classi e i progetti di Selenium in modo più semantico e farli funzionare.

Continuare con la documentazione.

# GESTIONE CASINÒ | Diario di lavoro - 17.05.2019

Matan Davidi, Thor Dublin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

Trevano, 17 maggio 2019

## Lavori svolti

Oggi Mattia ha sistemato il bottone modifica password della pagina di profilo:

# Modifica Password:

Email: carlo.pezzotti@samtrevano.ch

Modifica password

Cliccando questo pulsante l'utente verrà portato ad una pagina di conferma di modifica password. Questo è stato fatto con il seguente codice:

```
<form action="modifyPassword.php" method="post">
 <input type="submit" id="modify-password" class="form-control" value="Modifica password">
</form>
```

La pagina di conferma di modifica password è molto simile a quella di password smarrita, con l'unica differenza che la mail inserita non è modificabile e presa tramite il metodo POST. La pagina di modifica password si presenta così:



# Cambia Password

## Cambia la tua password!

Appena cliccherai INVIA controlla la tua email e clicca il link di modifica.

Email:

carlo.pezzotti@samtrevano.ch

INVIA

Infine ha continuato la parte di implementazione della documentazione, più precisamente la parte della pagina di profilo e di gestione delle pagine.

Matan oggi si è occupato di rinominare e in seguito creare i diagrammi UML delle classi, scritte in Java, dei test di Selenium. Queste classi sono state rinominate in:

- LoginTest
- NavigationTest
- RegistrationTest
- UserUpdateTest

LoginTest
-URL: String = "http://cashyland.tk/"; -driver: WebDriver = null
~accedi() ~insertEmail(email: String) ~insertPassword(password: String) ~pressButton() ~pressUsersManagement() ~pressRoomsManagement() ~pressGamesManagement() ~pressPromotionsManagement() ~pressImagesManagement() ~test() +waitForMillis(millis: int)

<b>NavigationTest</b>
-URL: String = "http://cashyland.tk/" -driver: WebDriver = null
~home() ~accedi() ~registrati() ~giochi() ~sale() ~map() ~test() +waitForMillis(millis: int)

<b>RegistrationTest</b>
-URL: String = "http://cashyland.tk/" -driver: WebDriver = null -email: String = "seleniumtest" -endEmail: String = "@gmail.com" -name: WebElement -cognome: WebElement -dataNascita: WebElement -via: WebElement -noCivico: WebElement -cap: WebElement -citta: WebElement -telefono: WebElement -indirizzoEmail: WebElement -pass: WebElement -repass: WebElement
~accedi() ~registrati() ~insertName(name: String) ~insertSurname(surname: String) ~insertBirthday(birthday: String) ~insertAddress(address: String) ~insertHouseNumber(houseNumber: String) ~insertZipCode(zipCode: String) ~insertCity(city: String) ~insertPhoneNumber(phoneNumber: String) ~insertEmail(email: String) ~insertPassword(password: String) ~insertRePassword(repassword: String) ~pressButton() ~getEmail(): String ~clearInputs() ~testTextInputs(text: String, input: String) ~testBirthday(birthday: String) ~testHouseNumber(houseNumber: String) ~testZipCode(zipCode: String) ~testEmail(email: String) ~testPhoneNumber(phoneNumber: String) ~testPassword(password: String) ~test() +waitForMillis(millis: int)

UserUpdateTest
-URL: String = "http://cashyland.tk/" -driver: WebDriver = null -email: String = "seleniumtest" -endEmail: String = "@gmail.com" -name: WebElement -cognome: WebElement -dataNascita: WebElement -via: WebElement -noCivico: WebElement -cap: WebElement -citta: WebElement -telefono: WebElement
~accedi() ~insertEmail(email: String) ~insertPassword(password: String) ~pressLoginButton() ~pressUpdateButton() ~pressUpdateUserButton() ~insertName(name: String) ~insertSurname(surname: String) ~insertBirthday(birthday: String) ~insertAddress(address: String) ~insertHouseNumber(houseNumber: String) ~insertZipCode(zipCode: String) ~insertCity(city: String) ~insertPhoneNumber(phoneNumber: String) ~clearInputs() ~testTextInputs(text: String, input: String) ~testBirthday(birthday: String) ~testHouseNumber(houseNumber: String) ~testZipCode(zipCode: String) ~testEmail(email: String) ~testPhoneNumber(phoneNumber: String) ~test() +waitForMillis(millis: int)

Carlo ha creato la presentazione al cliente del sito web per tutta la durata della giornata di lavoro. Il risultato è visibile seguendo questo [link](../presentation/Gestione Casinò).

Oggi Thor è stato assente le prime 2 ore, le seconde due ore inizialmente si è occupato di dare un'occhiata ai test di Selenium che riscontravano un errore la lezione precedente (riguardante la connessione e il display di xvfb), successivamente si è occupato dei Test Case, capitolo 4.1 della documentazione.

Oggi Matteo ha provato a capire perché i test di Selenium non funzionano. Data l'assenza di Thor ha provato a seguire da solo una guida trovata su internet al seguente link:

```
https://dzone.com/articles/run-headless-selenium-tests-from-jenkins
```

Prima di fare un test con Jenkins ha però provato ad eseguire a mano lo script che il software avrebbe dovuto lanciare ed esso dava dei problemi ma dopo una brava ricerca ha scoperto che mancava un componente al server e lo ha quindi installato grazie alla seguente pagina con il relativo comando:

```
https://askubuntu.com/questions/1005623/libdbusmenu-glib-warning-unable-to-get-session-bus-failed-to-execute-child
sudo apt-get install dbus-x11
```

Fatto ciò ha dovuto eliminare il lavoro fatto da Thor (per ora solo commentato) nelle ultime lezioni perché esso causa problemi.

```
System.setProperty("webdriver.gecko.driver", "/usr/bin/geckodriver");
/*String Xport = System.getProperty("lmportal.xvfb.id", ":0");

driver = new FirefoxDriver(new GeckoDriverService.Builder()
 .usingDriverExecutable(new File("/usr/bin/geckodriver"))
 .usingFirefoxBinary(new FirefoxBinary(firefoxPath))
 .withEnvironment(ImmutableMap.of("DISPLAY", Xport)).build());*/
driver = new FirefoxDriver();
```

Per lanciare i test ha poi seguito la seguente guida:

```
http://elementalseelenium.com/tips/38-headless
```

Dopo questi passaggi i test hanno iniziato, in parte, a funzionare. Essi infatti riescono a caricare la pagina iniziale ma non riescono a trovare gli elementi di essa. Per cercare di risolvere il problema Matteo ha provato ad aggiungere il seguente codice che dovrebbe bloccare il test fino a che la pagina non sia completamente caricata.

```
WebDriverWait wait = new WebDriverWait(driver, timeOutInSeconds: 20);
wait.until(ExpectedConditions.elementToBeClickable(By.className("container")));
```

Questo non ha comunque risolto i problemi.

## Problemi riscontrati e soluzioni adottate

---

### Punto della situazione rispetto alla pianificazione

---

Rispetto alla pianificazione siamo in orario.

### Programma di massima per la prossima giornata di lavoro

---

Continuare con la documentazione.

# GESTIONE CASINÒ | Diario di lavoro - 22.05.2019

---

Matan Davidi, Thor Düblin, Matteo Forni, Carlo Pezzotti, Mattia Toscanelli

**Trevano, 22 maggio 2019**

## Lavori svolti

---

Oggi l'intero gruppo si è occupato di completare la documentazione e le funzionalità della piattaforma.

Matan ha aggiunto tutte le didascalie alle immagini. In seguito ha aggiunto tutta la parte di implementazione riguardante la parte del database e ha corretto tutti gli errori di ortografia.

Matteo ha modificato lo stile della guida all'installazione e configurazione di Jenkins così da renderlo uguale alla documentazione principale. Inoltre ha aggiunto la parte di implementazione dei test di Selenium alla documentazione principale.

Mattia oggi ha sistemato i bug rimanenti nel sito e ha verificato che il sito funzionasse alla perfezione. Inoltre si è occupato di creare un documento unendo tutti i diari assieme alla documentazione.

Thor oggi si è dedicato alla documentazione, più precisamente ha scritto sotto il capitolo implementazione la parte riguardante l'installazione di selenium e il suo utilizzo. Inoltre ha scritto i test case.

## Problemi riscontrati e soluzioni adottate

---

## Punto della situazione rispetto alla pianificazione

---

Rispetto alla pianificazione siamo in orario.

## Programma di massima per la prossima giornata di lavoro

---