

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 Döblin, 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

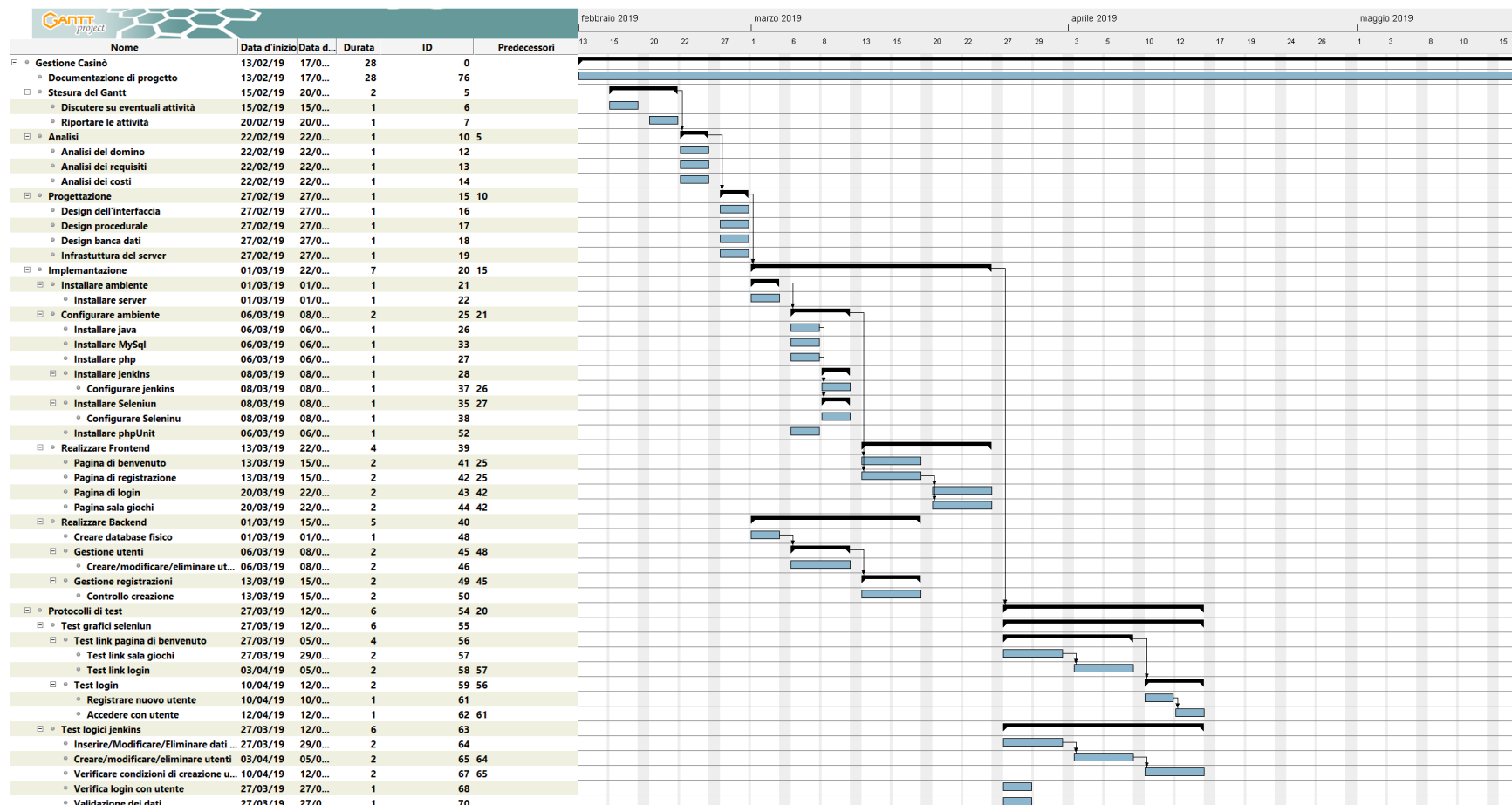


Figura 1: diagramm di Gantt utilizzato per pianificare il progetto

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.

1.6.1 Stesura del Gantt

☐ • Stesura del Gantt	15/02/19	20/0...	2	5	
• Discutere su eventuali attività	15/02/19	15/0...	1	6	
• Riportare le attività	20/02/19	20/0...	1	7	

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

☐ • Analisi	22/02/19	22/0...	1	10 5	
• Analisi del domino	22/02/19	22/0...	1	12	
• Analisi dei requisiti	22/02/19	22/0...	1	13	
• Analisi dei costi	22/02/19	22/0...	1	14	

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.

1.6.3 Progettazione

☐ • Progettazione	27/02/19	27/0...	1	15 10						
• Design dell'interfaccia	27/02/19	27/0...	1	16						
• Design procedurale	27/02/19	27/0...	1	17						
• Design banca dati	27/02/19	27/0...	1	18						
• Infrastruttura 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.

1.6.4 Implementazione

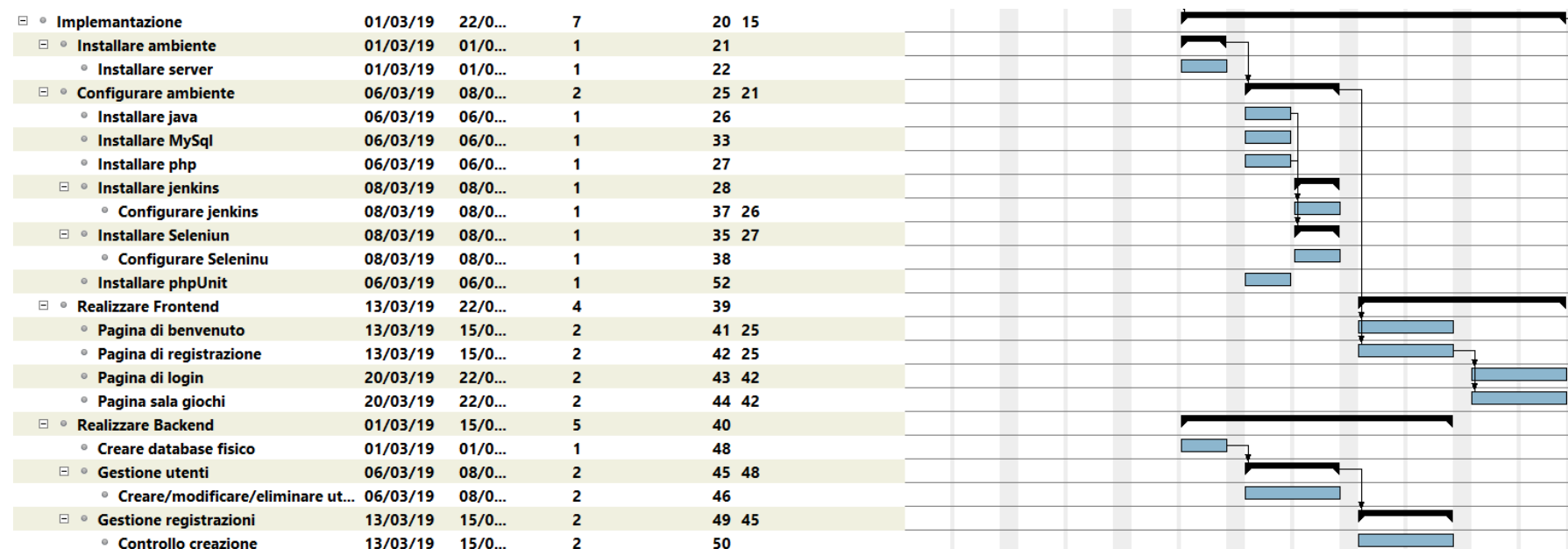


Figura 5: Attività "Implementazione"

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:



- 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


1.6.5 Protocolli di test

▢ • Protocolli di test	27/03/19	12/0...	6	54	20
▢ • Test grafici selenium	27/03/19	12/0...	6	55	
▢ • Test link pagina di benvenuto	27/03/19	05/0...	4	56	
• Test link sala giochi	27/03/19	29/0...	2	57	
• Test link login	03/04/19	05/0...	2	58	57
▢ • Test login	10/04/19	12/0...	2	59	56
• Registrare nuovo utente	10/04/19	10/0...	1	61	
• Accedere con utente	12/04/19	12/0...	1	62	61
▢ • Test logici jenkins	27/03/19	12/0...	6	63	
• Inserire/Modificare/Eliminare dati ...	27/03/19	29/0...	2	64	
• Creare/modificare/eliminare utenti	03/04/19	05/0...	2	65	64
• Verificare condizioni di creazione u....	10/04/19	12/0...	2	67	65
• Verifica login con utente	27/03/19	27/0...	1	68	
• Validazione dei dati	27/03/19	27/0...	1	70	

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.

	SAMT – Sezione Informatica	Pagina 13 di 54
	Gestione casinò	

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

2 Progettazione

2.1 Design dell'architettura del sistema

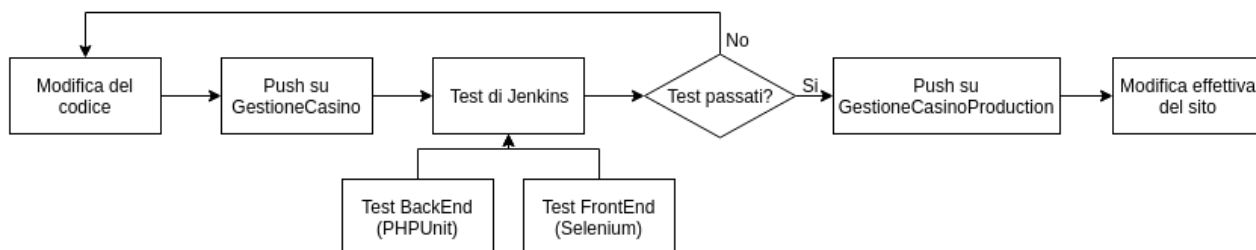


Figura 7 - Design dell'architettura del sistema

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:

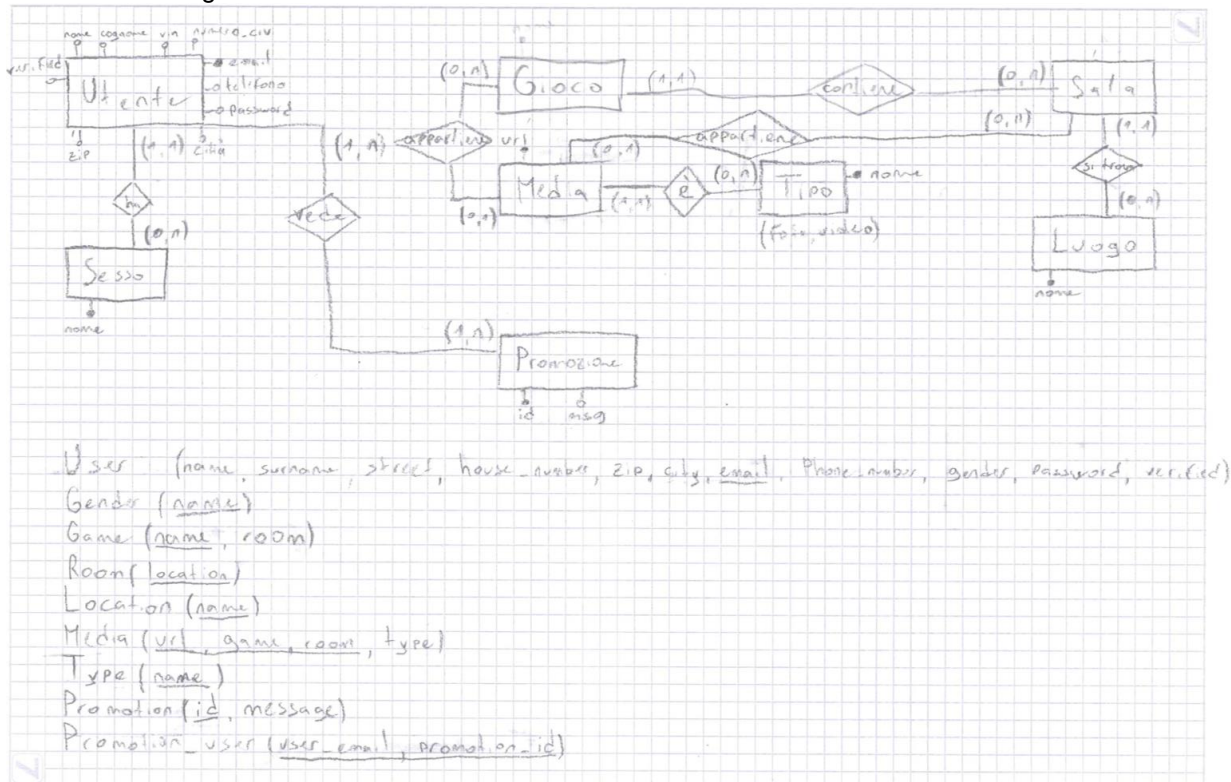


Figura 8 - Diagramma E/R usato per la progettazione del database

Come prima cosa, a sinistra troviamo la tabella dell'**Utente**, che viene identificato 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
-\$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()

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

2.3.1.2 DatabaseTestCase

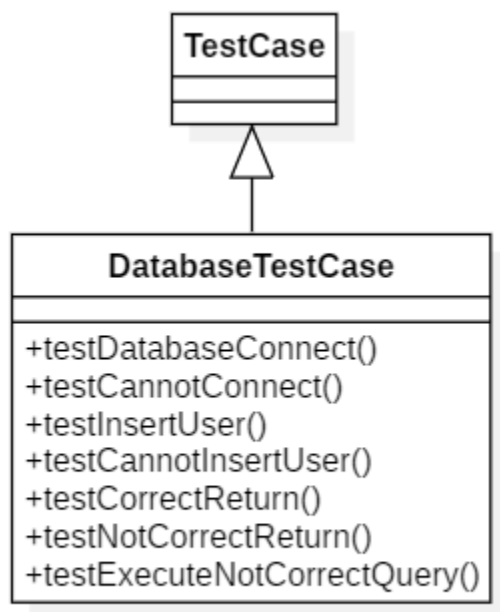


Figura 10 - Diagramma UML utilizzato per la progettazione della classe DatabaseTestCase

2.3.1.3 SendMailTest

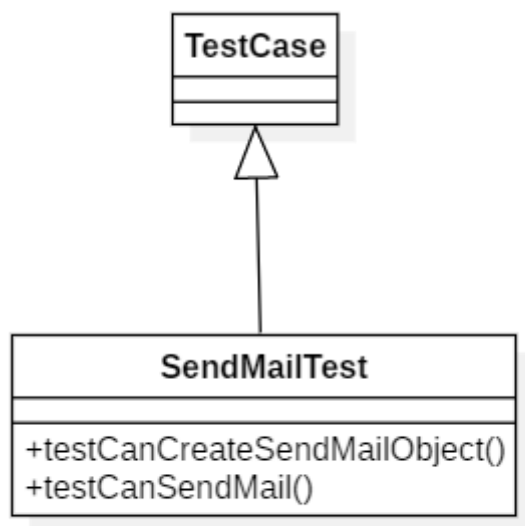


Figura 11 - Diagramma UML utilizzato per la progettazione della classe SendMailTest

2.3.1.4 UserTestCase

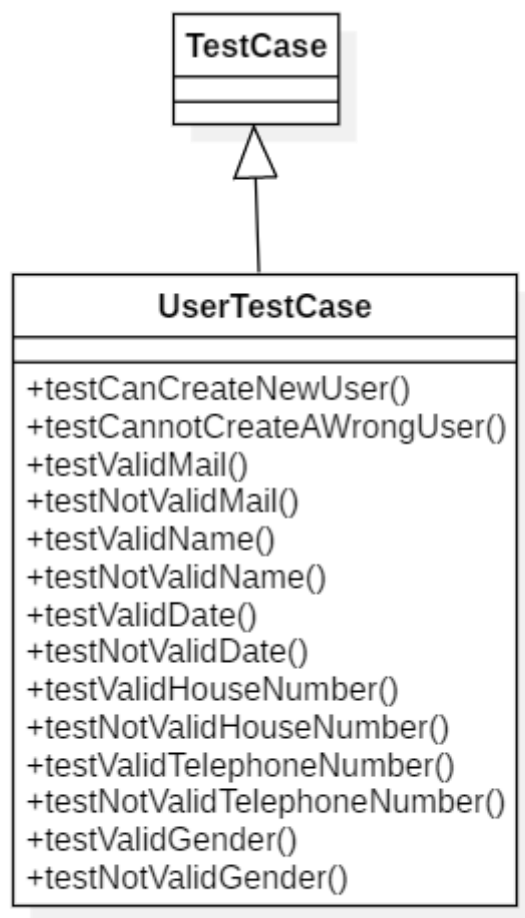


Figura 12 - Diagramma UML utilizzato per la progettazione della classe UserTestCase

2.3.2 Java

2.3.2.1 LoginTest

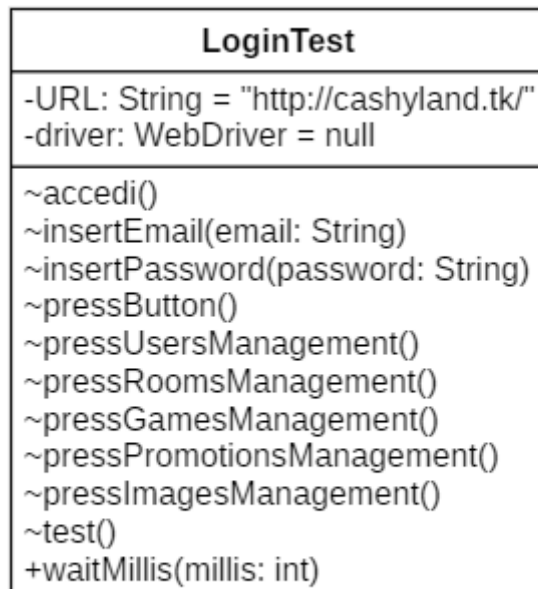


Figura 13 - Diagramma UML utilizzato per la progettazione della classe LoginTest

2.3.2.2 NavigationTest

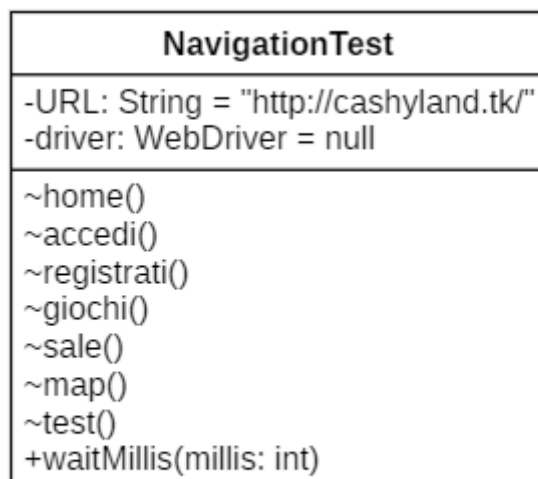


Figura 14 - Diagramma UML utilizzato per la progettazione della classe NavigationTest

2.3.2.3 RegistrationTest

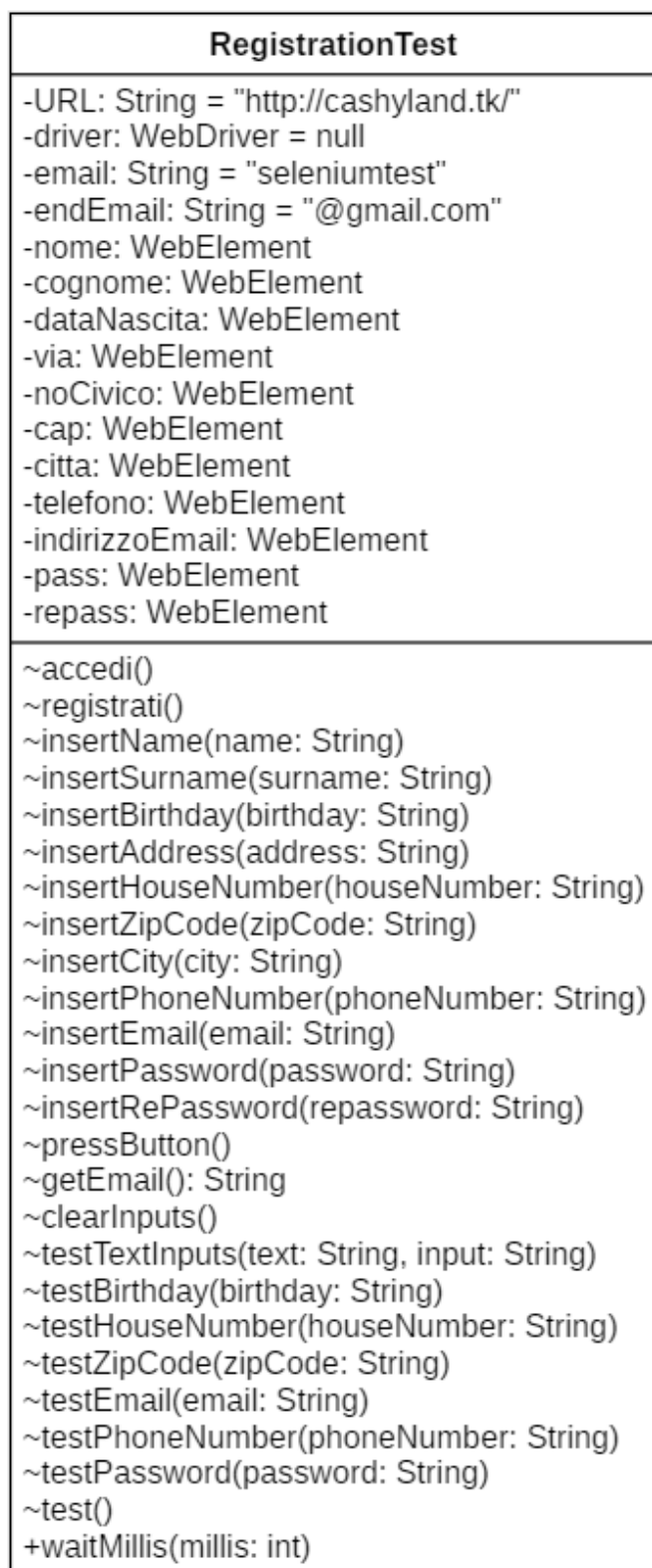


Figura 15 - Diagramma UML utilizzato per la progettazione della classe RegistrationTest

2.3.2.4 UserUpdateTest

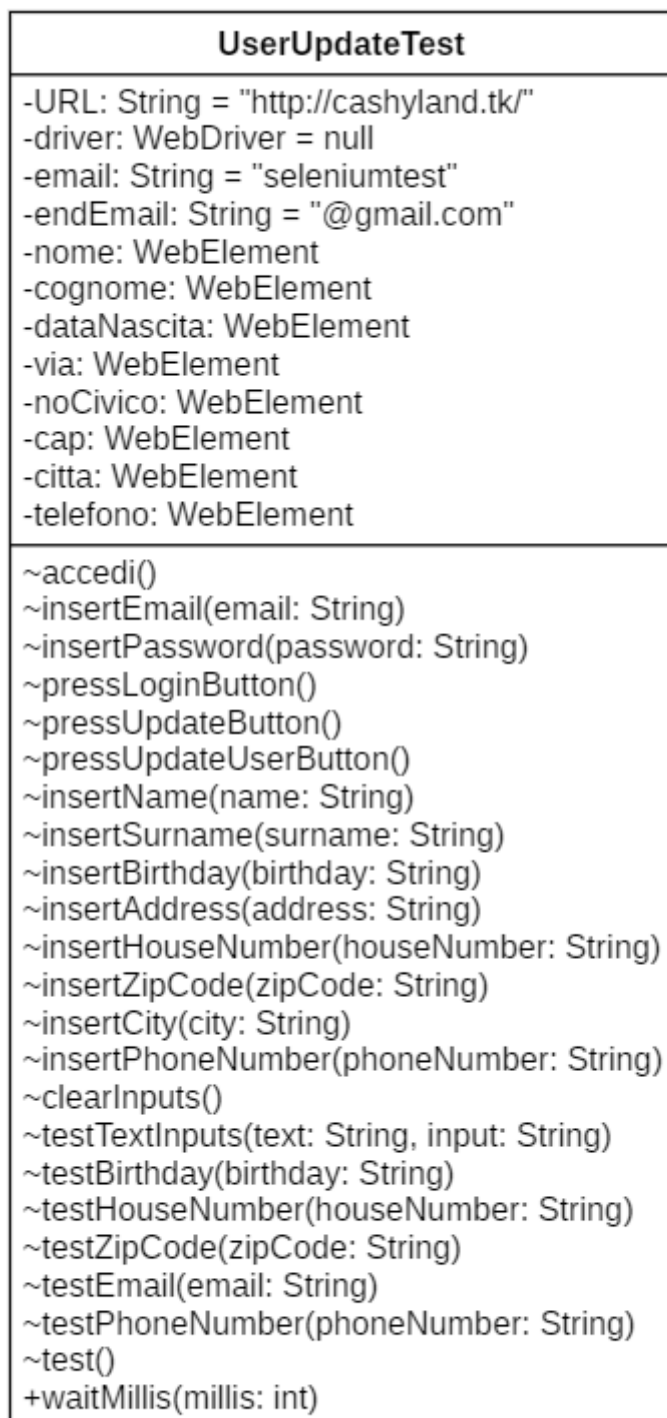


Figura 16 - Diagramma UML utilizzato per la progettazione della classe UserUpdateTest

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:

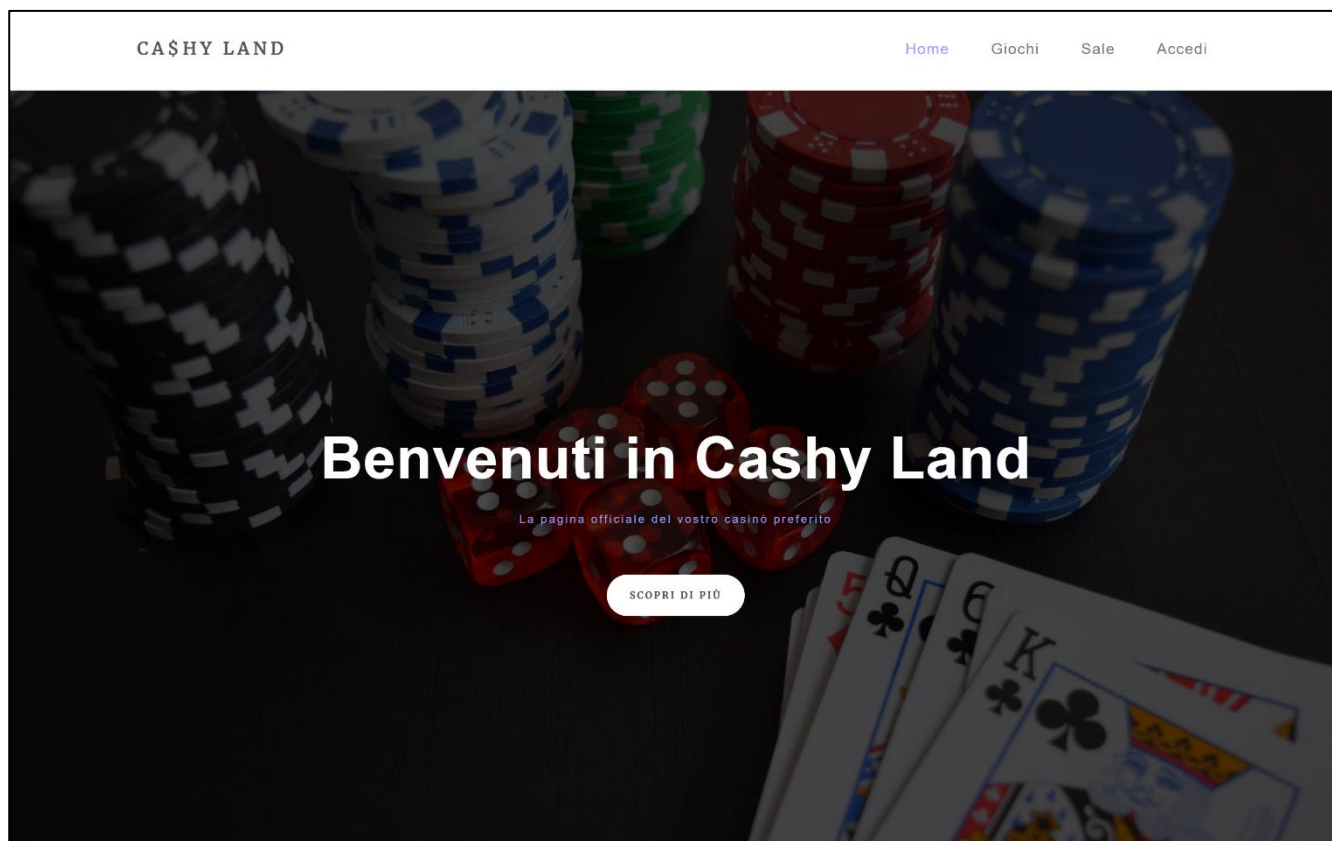


Figura 17 - Pagina di benvenuto 1

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:

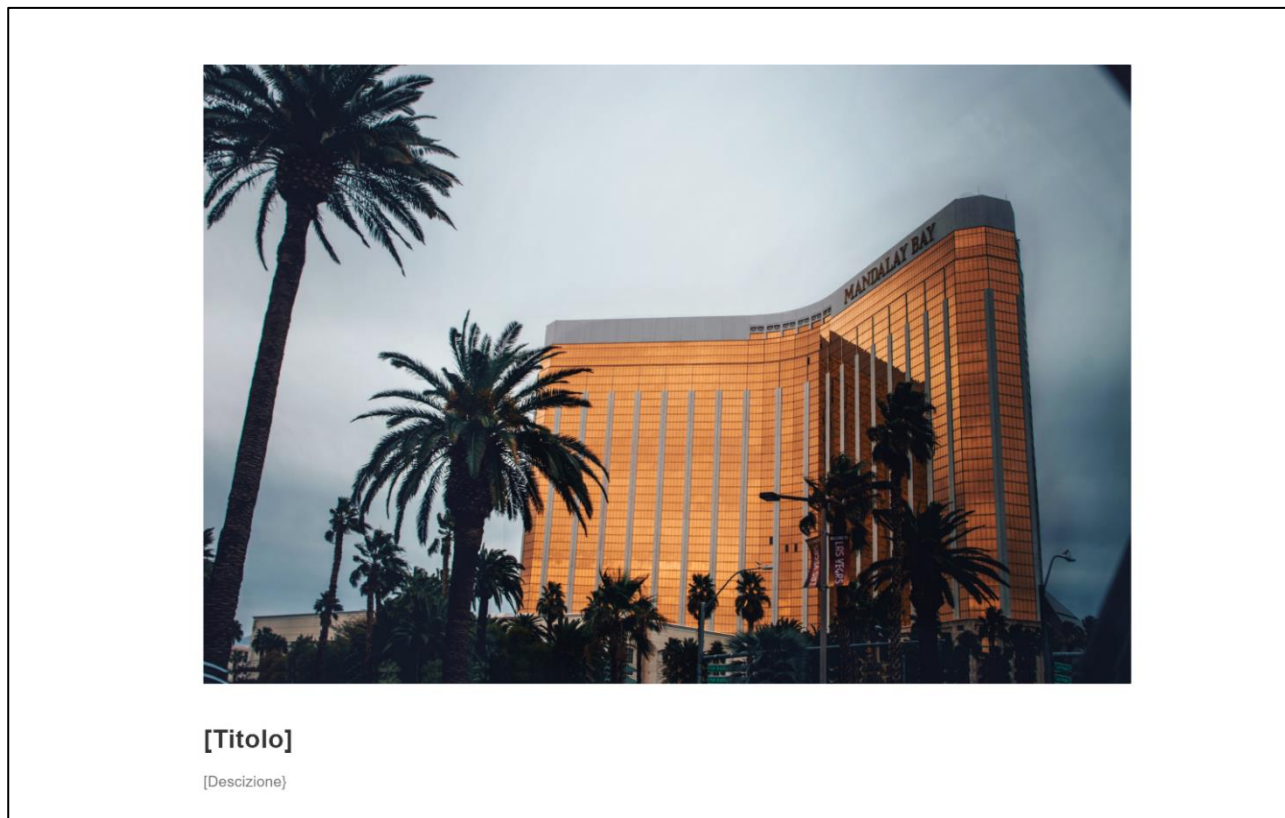
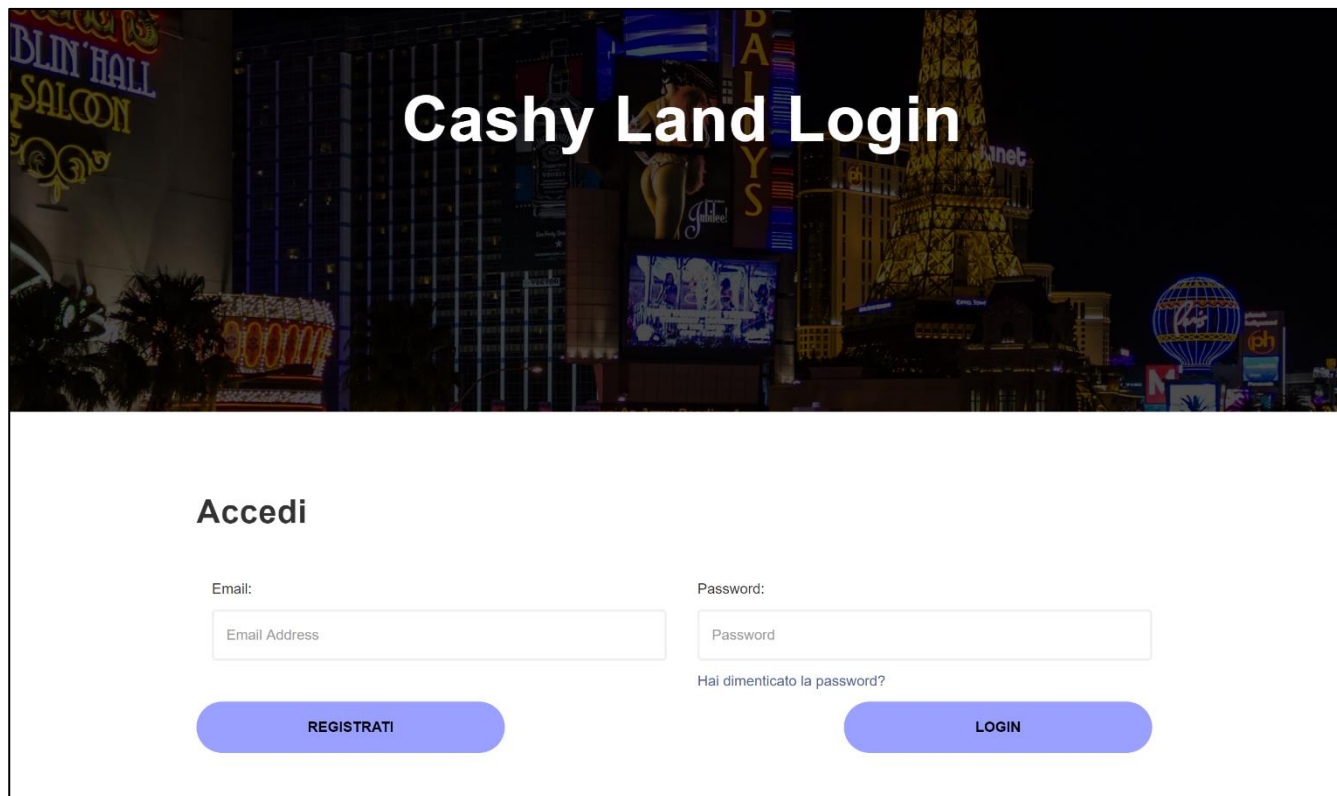


Figura 18 - Pagina di benvenuto 2

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.



Cashy Land Login

Accedi

Email:

Password:

[Hai dimenticato la password?](#)

REGISTRATI **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:

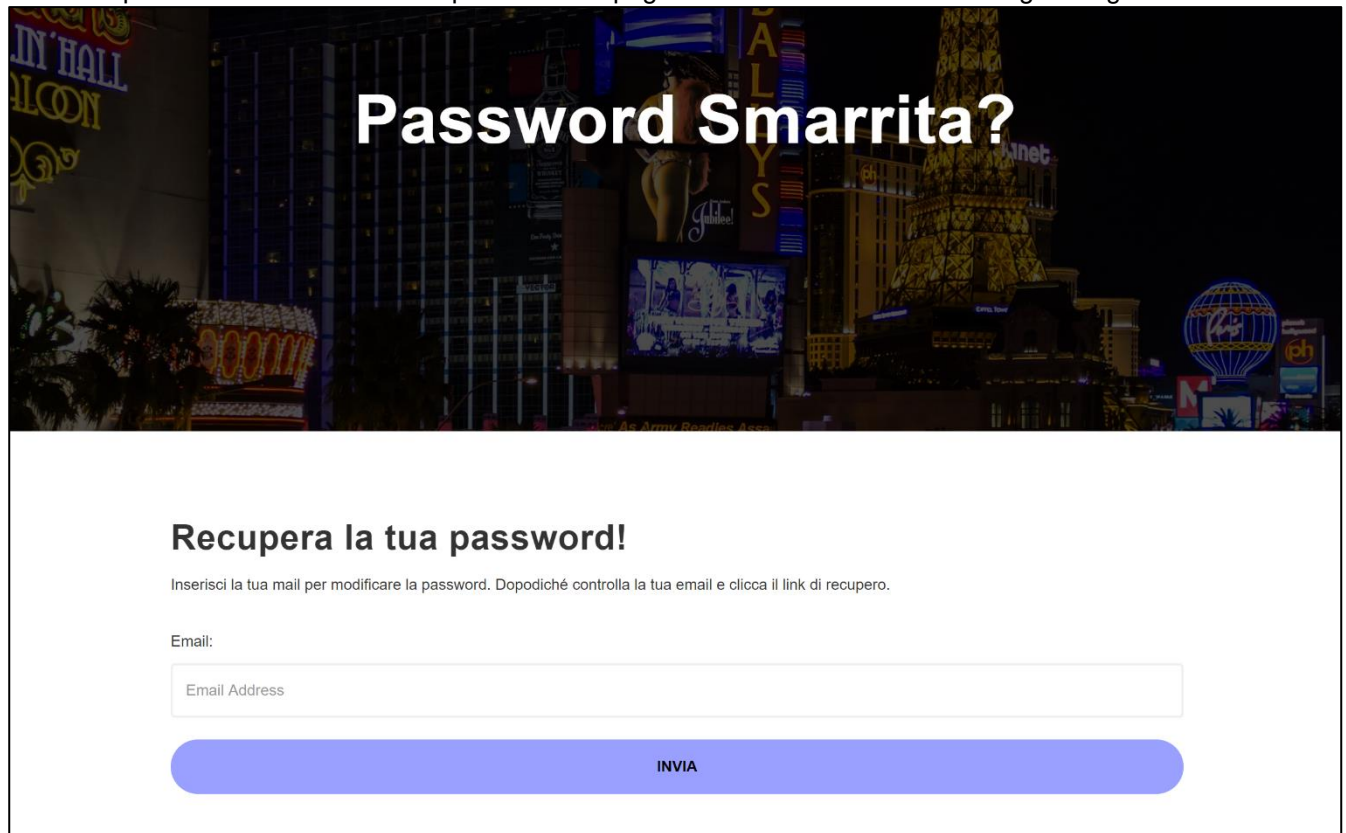


Figura 20 - Pagina di recupero password

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:

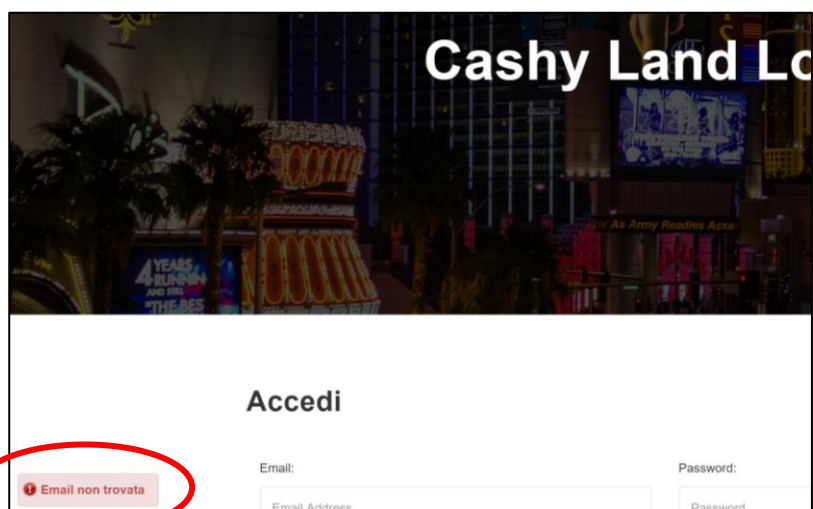



Figura 21 - Pagina di accesso 2

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:

Data Nascita:

Via:

No. Civico:

CAP:

Città:

No. Telefono:

Sesso:

Email:

Password:

Ripeti Password:

REGISTRATI

Figura 22 - Pagina di registrazione 1

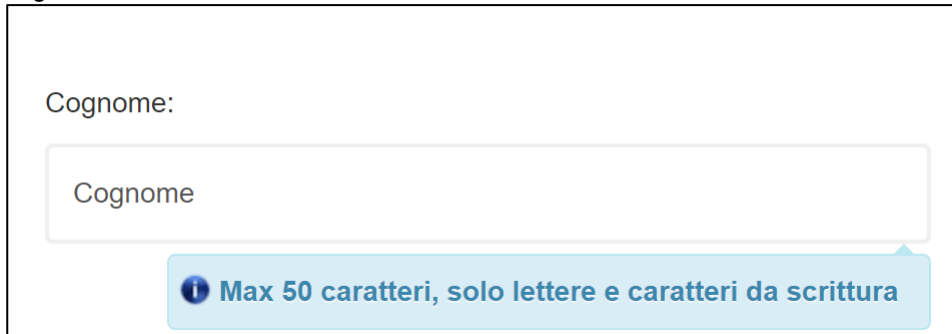
La pagina di registrazione è formata da un form in HTML costituita da 12 campi input. Tutti campi sono obbligatori, quindi se uno di quest'ultimi non viene compilato verrà mostrata sullo schermo una notifica di errore mostrando come bisogna compilarlo. Inoltre il campo input errato verrà contrassegnato 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:

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:

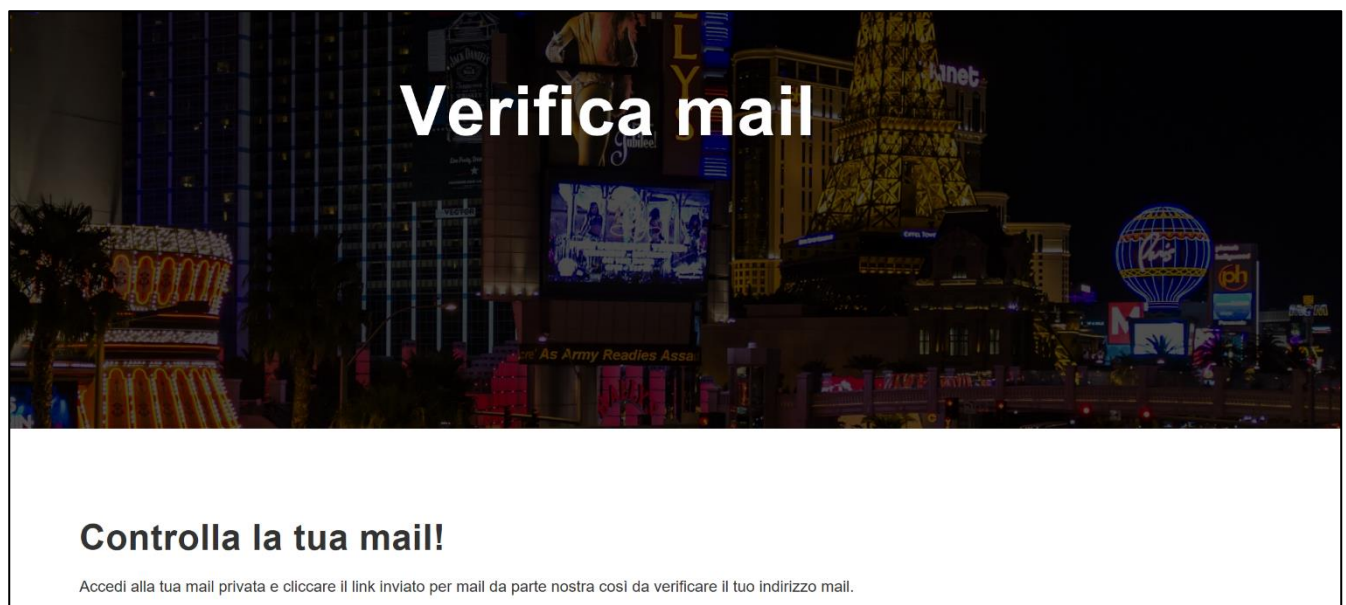


Figura 25 - Registrazione completata

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:

Email: prova@yopmail.com

[Modifica password](#)

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:

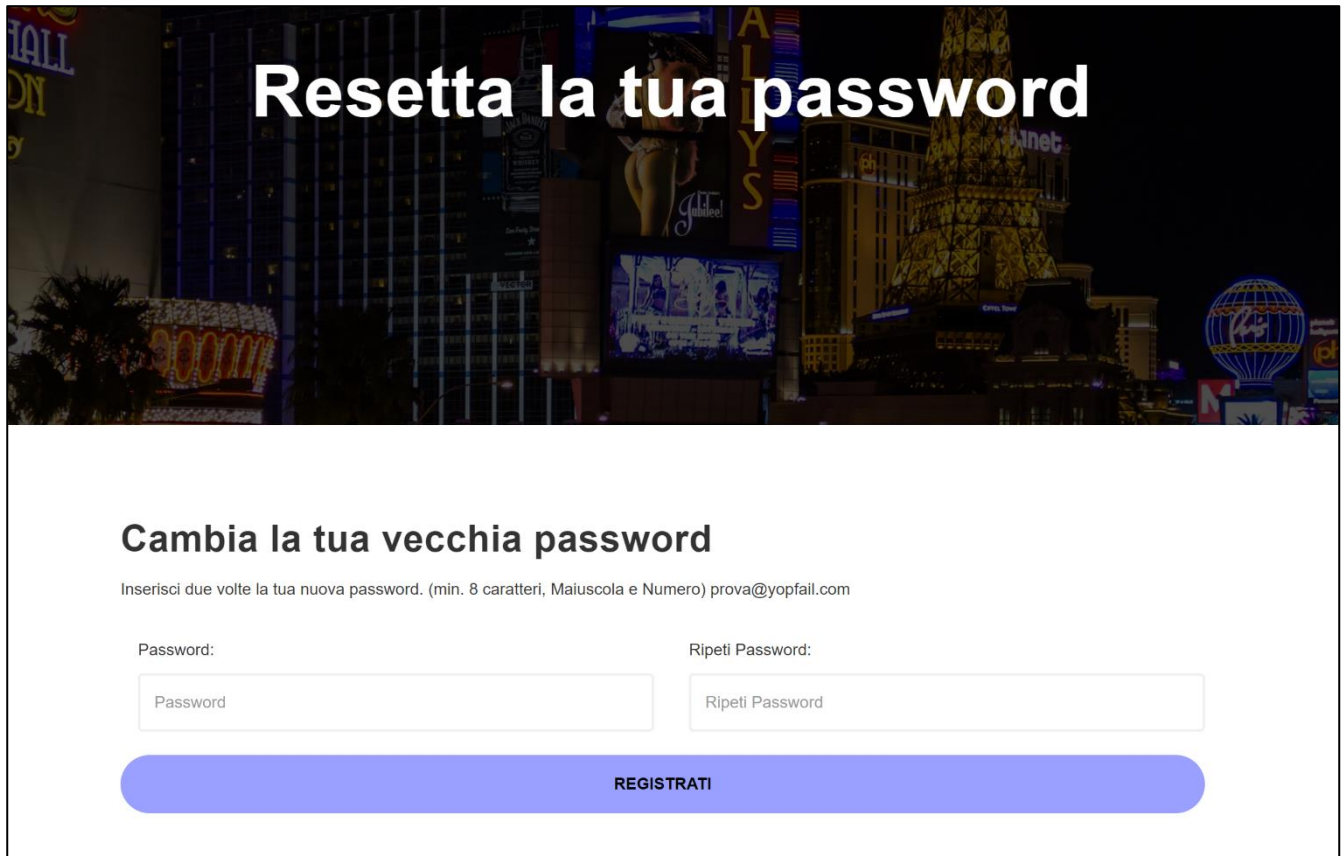


Figura 27 - Pagina di recupero password 2

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 e 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 = "#ffc000";
            inputs[1].style.backgroundColor = "#ffc000";
        }
    }
}
```



```
$.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
 	<u>Primo Piano</u>	In questa sala si possono trovare diversi tavoli da gioco per il Black Jack. Non resta altro che andare a provare a vin
 	<u>Secondo Piano</u>	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, $e
mail, $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 lancio una eccezione con un messaggio personalizzato.

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

```

    }else{
        throw new IllegalArgumentException(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 istanzia 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 istanzia 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>";
```

```

        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 inviargli 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 una 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);
$cryptedMail = $_POST["email"] ^ $privateKey;
$message = '<hr>How are you? <br> <hr>This is your link:<a
href="http://cashyland.tk/php/login/validate.php?id='.urlencode($cryptedMail).'">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>
</project>
```

```
</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 IntelliJ 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 -xvzf 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("portal.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 dilettarsi 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_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:	Dalla Pagina principale del sito (Home) seguire i passi descritti di seguito: <ol style="list-style-type: none"> 1. Cliccare sul link "Accedi", che porta alla pagina di Login. 2. Premere il pulsante "REGISTRATI", che porta alla pagina di Registrazione. 3. Selezionare il link "Accedi", che porta alla pagina di Login. 4. Seguire il link "Hai dimenticato la password?", che porta alla pagina della password smarrita. 5. Premere il link "Accedi", che porta alla pagina di Login. 6. Selezionare il link "Home", che porta alla pagina principale. 7. Cliccare sul link "Giochi", che porta alla pagina dei giochi. 8. Premere il link "Home", che porta alla pagina principale. 9. Seguire il link "Sale", che porta alla pagina delle sale. 10. 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.		

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 “1234567890123456789012345678901234567890123456789012345678901” 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". <p>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):</p> <ol style="list-style-type: none"> 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).		

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 valido aspettandosi che quest'ultimo tiri un exception 5. Provare a richiamare il metodo che definisce l'età in base a una data passate. 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.		

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

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

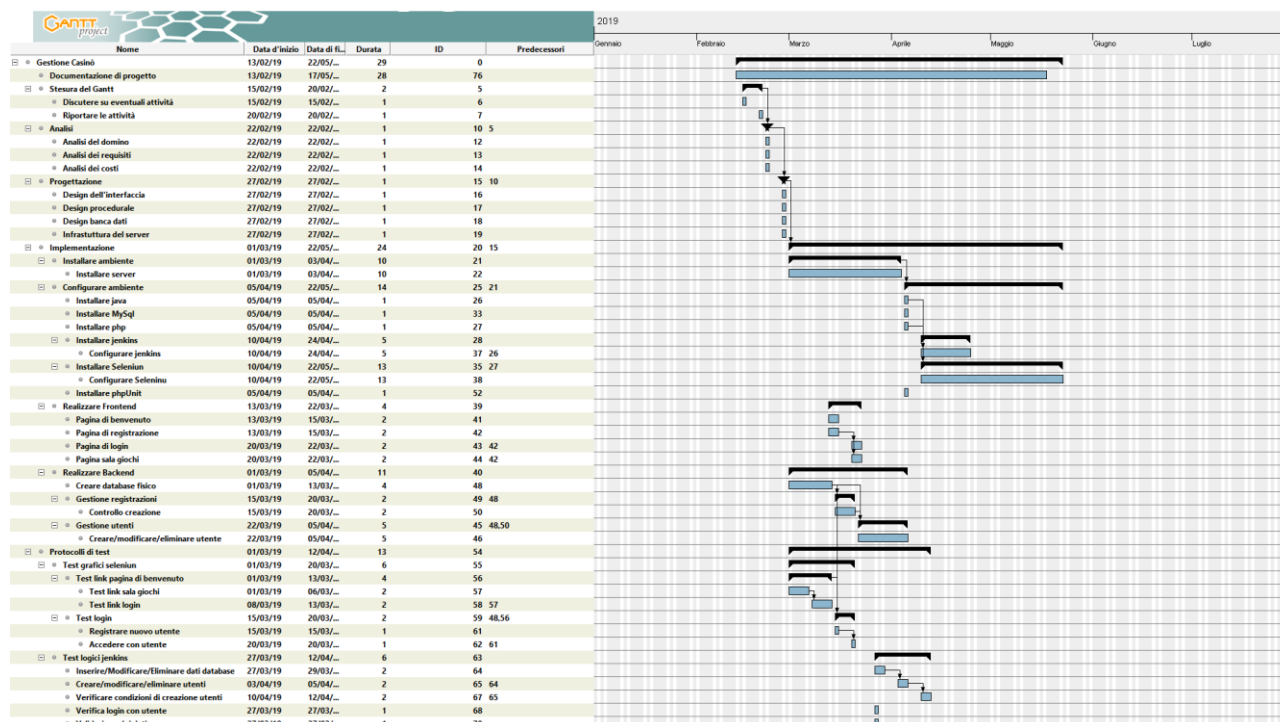


Figura 31 - Diagramma di Gantt consuntivo

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