Gestione casinò

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

Analisi 4

1.4 Analisi del dominio 4

1.5 Analisi e specifica dei requisiti 4

1.6 Use case 6

1.7 Pianificazione 6

1.8 Analisi dei mezzi 6

1.8.1 Software 6

1.8.2 Hardware 6

2 Progettazione 7

2.1 Design dell’architettura del sistema 7

2.2 Design dei dati e database 7

2.3 Design delle interfacce 7

2.4 Design procedurale 7

3 Implementazione 8

4 Test 8

4.1 Protocollo di test 8

4.2 Risultati test 9

4.3 Mancanze/limitazioni conosciute 9

5 Consuntivo 9

6 Conclusioni 9

6.1 Sviluppi futuri 9

6.2 Considerazioni personali 9

7 Bibliografia 9

7.1 Bibliografia per articoli di riviste: 9

7.2 Bibliografia per libri 9

7.3 Sitografia 9

8 Allegati 10

# Introduzione

## Informazioni sul progetto

Autore: Matan Davidi, Thor Düblin, 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

## 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)

## 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

## 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.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| Nome | Sito web per la gestione di un casinò |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| Nome | Gestione degli utenti |
| Priorità | 2 |
| Versione | 1.0 |
| Note |  |
| *Sotto-requisiti* | |
| 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 |  |
| *Sotto-requisiti* | |
| 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 |  |
| *Sotto-requisiti* | |
| 001 | Per ogni sala deve essere gestita la posizione all’interno casinò. |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| Nome | Gestione promozioni |
| Priorità | 2 |
| Versione | 1.0 |
| Note |  |
| *Sotto-requisiti* | |
| 001 | Devono essere gestite delle promozioni da mostrare a uno o più utenti sotto forma di messaggi pop-up. |
| 002 | Ogni promozione deve contenere un messaggio da mostrare a un utente |
| 003 | Deve essere gestito quale promozione viene visualizzata da quali utenti |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| Nome | Protocolli di test e integrazione continua |
| Priorità | 2 |
| Versione | 1.0 |
| Note |  |
| *Sotto-requisiti* | |
| 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 |

## 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.

### 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.

### 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.

### Progettazione

****

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.

### 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:
  + 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:
  + Installare Java
  + Installare MySQL
  + Installare PHP
  + 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.
  + Installare Selenium
    - Configurare Selenium, per fargli eseguire i test dell’interfaccia grafica su “ordine” di Jenkins dopo ogni push.
  + 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

### 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](https://www.seleniumhq.org/), 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 di 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.

### Documentazione di progetto

Infine c’è un’attività rimasta costant 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.

## Analisi dei mezzi

### 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

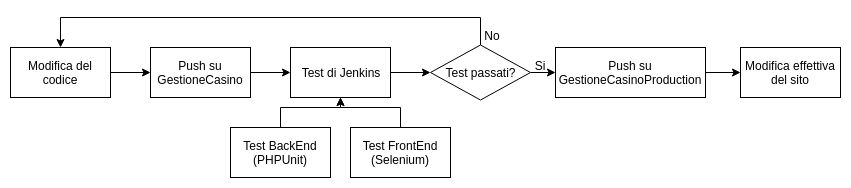
### Hardware

Macchina server:

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

# Progettazione

## Design dell’architettura del sistema



## 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:



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)

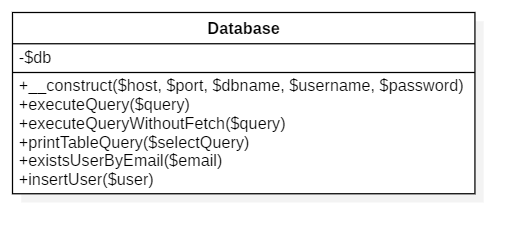
## Design procedurale

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

### PHP

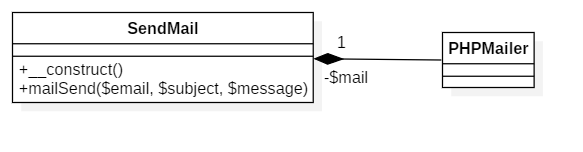
#### Database

La classe Database contiene i metodi utili a instaurare una connessione con il database MySQL su cui si basa il progetto. Le operazioni che permette di fare sono: ottenere il risultato di una query (risultato sotto forma di array con executeQuery o di oggetto di tipo mysqli\_result con executeQueryWithoutFetch), stampare una tabella HTML contenente il risultato di una query, controllare se un utente esiste tramite la sua email e inserire un nuovo utente.



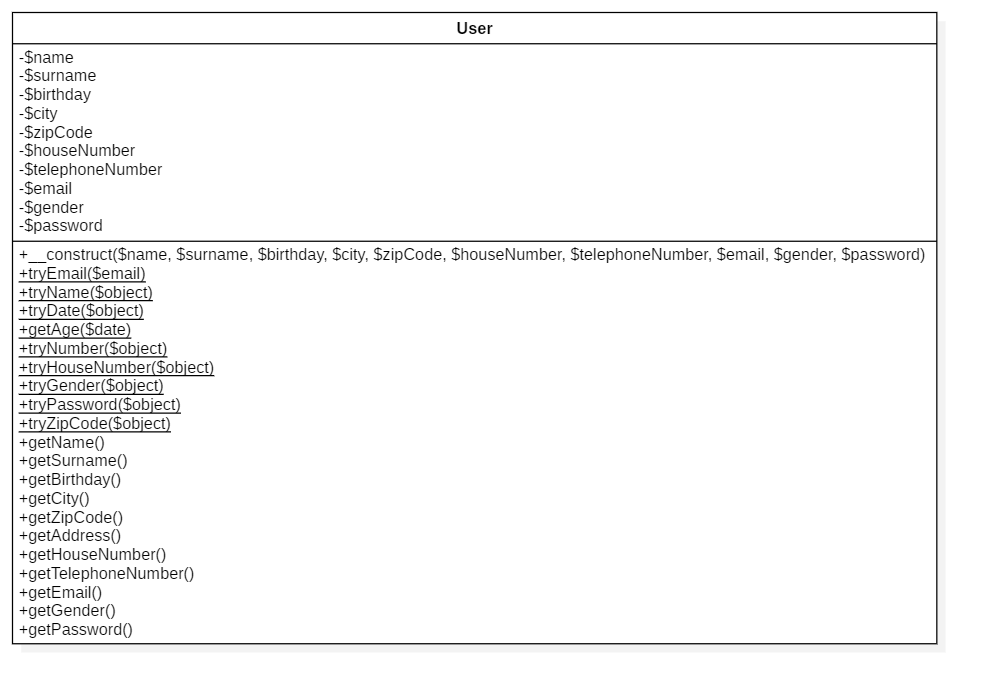
#### SendMail

La classe SendMail permette di inviare un’email ad un certo indirizzo di posta elettronica. Per farlo si avvale della classe di libreria PHPMailer, grazie all’istanza contenuta nella variabile $mail.



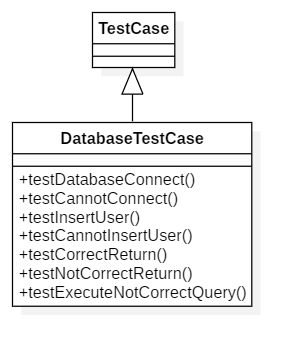
#### User

La classe sulla quale si basano i test relativi agli utenti. Questa classe contiene un attributo per ogni campo del form di registrazione, un metodo costruttore e un metodo per ogni attributo che verifichi che sia valido, quindi contiene:



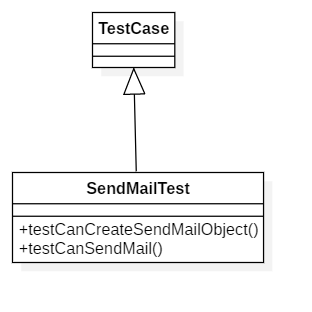
#### DatabaseTestCase

DatabaseTestCase è una classe di test che permette di verificare il corretto funzionamento della connessione al database e della lettura e scrittura all’interno di esso. Per fare ciò, la classe contiene due metodi per ognuna di queste azioni, uno per verificare che funzioni se eseguita correttamente e uno per accertarsi che non funzioni se invece non viene svolta nel modo giusto.



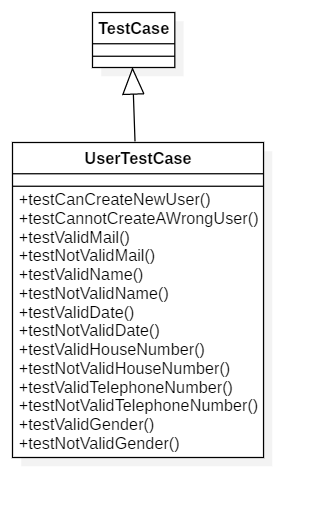
#### SendMailTest

La classe SendMailTest contiene la logica per accertarsi che sia possibile creare e inviare correttamente un’email a un indirizzo di posta elettronica. Contiene, quindi, un metodo che verifichi di poter creare un oggetto di tipo SendMail e un metodo che si accerti di poterla inviare.



#### UserTestCase

La classe di test UserTestCase ha al suo interno i metodi utilizzati per testare la validità dei dati inseriti all’interno del form di registrazione e per controllare se è possibile creare un’istanza di User con dei dati validi e non sia possibile con dei dati non validi.



### Java

# Implementazione

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

In questa parte è richiesto l’inserimento di codice sorgente/print screen di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

In base al browser web con il quale si visita il sito, alcune funzionalità cominciano o smettono di funzionare:

* Se si accede alla pagina usando Google Chrome, il bottone “Scopri di più” non porta più in basso nella pagina.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap 1.7) (ad esempio Gannt consuntivo).

# 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.

## 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.

## 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.

# Bibliografia

## 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 &#39;munin&#39; 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 &amp; 16.04 &ndash; TecAdmin, 17.04.2019
* <https://gist.github.com/ziadoz/3e8ab7e944d02fe872c3454d17af31a5>, Install Chrome, ChromeDriver and Selenium on Ubuntu 16.04 · GitHub, 17.04.2019

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o Qdc
* Prodotto
* …