

Politecnico di Milano

Facoltà di Ingegneria Industriale e dell’Informazione

Corso di Laurea Triennale in Ingegneria Informatica

Immagine che contiene testo, Carattere, cerchio, schermata

Descrizione generata automaticamente

Candidato:

Nome: **Ettore Cirillo Mugisha**

Matricola: **991290**

Codice Persona: **10864944**

Relatore: Professore **Salnitri Mattia**

INDICE

[**1 - BLACK JACK SAFER** 3](#_Toc167188901)

[1.1 - Descizione della Tematica 3](#_Toc167188902)

[1.2 - Obiettivo 3](#_Toc167188903)

[1.3 - Analisi dei Requisiti 3](#_Toc167188904)

[1.3.1 - Requisiti Funzionali 3](#_Toc167188905)

[1.3.1.1 - Use Case Diagrams 3](#_Toc167188906)

[1.3.2 - Requisiti non funzionali 4](#_Toc167188907)

[1.3.3 - Design Strutturale 4](#_Toc167188908)

[**2 - Database** 5](#_Toc167188909)

[2.1 - Diagramma ER 5](#_Toc167188910)

[2.2 - Annotazioni Tecniche 5](#_Toc167188911)

[**3 - Back-End** 5](#_Toc167188912)

[3.1 - Progettazione Architetturale 5](#_Toc167188913)

[3 1.1 - Pattern MVCS 5](#_Toc167188914)

[**4 - Front-End** 6](#_Toc167188915)

[4.1 - Tecnologie Utilizzate 6](#_Toc167188916)

[4.1.1 - Angular 6](#_Toc167188917)

[4.1.1.1 - Componenti 6](#_Toc167188918)

[4.1.1.2 - Services 7](#_Toc167188919)

[4.1.1.3 Routes 7](#_Toc167188920)

# **1 - BLACK JACK SAFER**

## 1.1 - Descizione della Tematica

Il gioco d’azzardo, fin dai tempi più antichi, è sempre stato un’attività che ha caratterizzato l’esperienza umana, spesso utilizzato come valvola di sfogo per emozioni intense o come strumento per indurre eccitazione ed adrenalina nella vita quotidiana. Da esso nasce lo studio della probabilità e della statistica, disciplina di grande studio al giorno d’oggi.

Tra i giochi d’azzardo, il BlackJack si distingue per la sua intuitività ed accessibilità. Notoriamente riconosciuto per la possibilità di applicare strategie matematiche, il BlackJack è considerato da molti il gioco d’azzardo più “sicuro”. Tale percezione, deriva dalla capacità di ridurre il vantaggio del banco attraverso decisioni strategiche informate, basate su calcoli matematici esatti.

## 1.2 - Obiettivo

“BlackJack Safer” nasce con l’intento di trasformare l’approccio al gioco d’azzardo online, puntando alla riduzione dei rischi di dipendenza e all’educazione di essi alla strategia migliore. Questo obiettivo è perseguito attraverso una gestione più umana e responsabile del saldo degli utenti, con un sistema che prevede l’approvazione (o il rifiuto) per ogni operazione di ricarica. In questo modo, dunque, non solo si protegge l’utente dal rischio di comportamenti compulsivi, ma si propone anche di educarli ad un approccio al gioco più consapevole e controllato.

Il software utilizza tecnologie moderne e strategie di intervento precoce per garantire un’esperienza di gioco equilibrata e sicura, mirando a mantenere il divertimento del BlackJack senza trascurare la salute ed il benessere dei giocatori.

## 1.3 - Analisi dei Requisiti

### 1.3.1 - Requisiti Funzionali

I requisiti funzionali rivestono un ruolo cruciale nel processo di sviluppo del software, in quanto costituiscono un pilastro per la progettazione, lo sviluppo, il controllo e la convalida. Documentare i requisiti funzionali contribuisce a garantire che il software risponda alle esigenze degli utenti e dell'organizzazione. Il progetto prevede l'interazione tra tre differenti tipi di utenti:

**ADMIN –** Agirà come supervisore dell’applicazione. Esso avrà la possibilità di: creare/Eliminare dei “Tabacchi” dove gli utenti potranno richiedere di ricaricare il saldo, Creare/Modificare/Eliminare gli Economi, un particolare tipo di utente i quali gestiranno le ricariche di denaro, Modificare/Eliminare gli utenti PLAYER.

**ECONOMO –** Agirà come gestore del saldo degli utenti Player. Esso, associato ad uno o più specifici Tabacchi, avrà la possibilità di accettare o rifiutare le richieste di ricarica mandate dagli utenti. L’esito della richiesta sarà notificato successivamente al mittente.

**PLAYER –** Agirà come giocatore effettivo del casinò. Esso avrà la possibilità di unirsi ad un tavolo e giocare la sua partita a blackJack fin quando vuole. Qualora occorre, avrà la possibilità di effettuare una richiesta di ricarica del saldo, la quale dovrà essere approvata dall’utente.

#### 1.3.1.1 - Use Case Diagrams

<Metti i diagrammi con immagini>

### 1.3.2 - Requisiti non funzionali

I requisiti non funzionali di una Web Application, se rispettano gli standard e le normative del settore, sono essenziali per garantire una user experience migliore possibile. Esse si focalizzano sulle performance, sulla qualità e la sicurezza. In questo progetto sono rese in evidenza le più rilevanti tra cui:

**Scalabilità –** Uno dei requisiti più importanti del software è quello di avere una struttura facile da mantenere, aggiornare ed estendere in futuro. Essa, infatti, pone le fondamenta per una manutenibilità più agevole e conveniente per gli sviluppatori futuri. Tra gli aspetti più importanti della scalabilità del software troviamo:

* **Leggibilità –** Scrivere il codice secondo le convezioni globali e rispettando le norme della programmazione è una buona prassi per rendere l’applicazione facile da capire per gli sviluppatori futuri.
* **Commenti e Documentazione -** I commenti, con anche l’uso della documentazione, velocizzano notevolmente il reverse engineering degli sviluppatori.
* **Versionamento –** L’uso di tool di controllo di versione del codice, come *Git*, permette agli sviluppatori di tenere traccia delle modifiche implementate sul software, potendo così navigare tra esse in maniera facile.
* **Testing –** Esso è fondamentale per verificare che l’applicazione funzioni correttamente, potendo così identificare casi limite e bug di sistema.

**Sicurezza –** pietra miliare nel mondo dell’IoT, essa è fondamentale per la protezione da minacce, pericoli o rischi che comprometterebbero l’integrità e la disponibilità dei dati e delle informazioni, come i dati sensibili degli utenti. Tra gli aspetti chiave della sicurezza troviamo:

* **Integrità –** Essa comprende la crittografia dei dati, l’implementazione di controlli di accesso e validazione dei dati in ingresso volti a garantire che i dati o il sistema non risultino alterati in modo anomalo.
* **Autenticazione e autorizzazione –** Per verificare l'identità di un utente e consentire l'accesso a risorse o funzionalità specifiche, utilizziamo il *JWT* (JSON Web Token). Questo token crittografato contiene dati sull'utente autenticato e può essere validato per assicurarne l'autenticità. Questo metodo fornisce un modo efficiente e sicuro per gestire l'accesso degli utenti, permettendo loro di accedere a determinate parti dell'applicazione solo se sono correttamente autenticati e autorizzati.

**Applicabilità –** Essa misura la facilità d’utilizzo con la quale l’utente prossimo riesce ad interagire con l’applicativo in modo efficace. Tra i vari aspetti chiave troviamo:

* **UX/UI:** L’interfaccia grafica dell’applicazione deve essere facile ed intuitiva per garantire che l’utente non abbia difficoltà a navigare nell’applicazione

### 1.3.3 - Design Strutturale

L’applicazione si struttura su 3 livelli; tale approccio, tipico della progettazione di Web Application, mira a suddividere l’applicazione in tre componenti distinti per garantire una disposizione ben organizzata e scalabile. Ogni livello svolge un ruolo preciso e si occupa di determinati aspetti all’interno dell’applicazione.

15 fundamental tips on REST API design | by Williams O | MediumImmagine che contiene testo, schermata, logo, automobile

Descrizione generata automaticamenteImmagine che contiene Elementi grafici, grafica, clipart, creatività

Descrizione generata automaticamenteImmagine che contiene simbolo

Descrizione generata automaticamente



**Client** – Esso avrà il compito di mostrare all’utente le informazioni da lui richieste in modo intuitivo, ottenute tramite richieste al server.

**Server** – Esso avrà il compito di fornire i servizi o le risorse richieste dal Client, avendo interazione con il Database.

**Database** – Esso rappresenta l’effettivo sistema di archiviazione dell’applicazione. Esso ha la possibilità di organizzare, memorizzare e recuperare dati in maniera dinamica.

# **2 - Database**

## 2.1 - Diagramma ER

<Metti l’immagine del giagramma ER>

## 2.2 - Annotazioni Tecniche

La scelta di utilizzare un *relational database management system* come PostreSQL è stata fatta per varie ragioni: poiché open source, esso ha alle spalle una comunità di sviluppatori che contribuiscono a mantenerlo; inoltre, esso possiede funzionalità performanti per quanto riguarda la **sicurezza** e la **geolocalizzazione**.

Per la tabella dell’utente, è stato scelto il nome “***\_user***” poiché “*user*” è una parola riservata in diversi DBMS, tra cui *Postgres*.

# **3 - Back-End**

## 3.1 - Progettazione Architetturale

### 3 1.1 - Pattern MVCS

# **4 - Front-End**

## 4.1 - Tecnologie Utilizzate

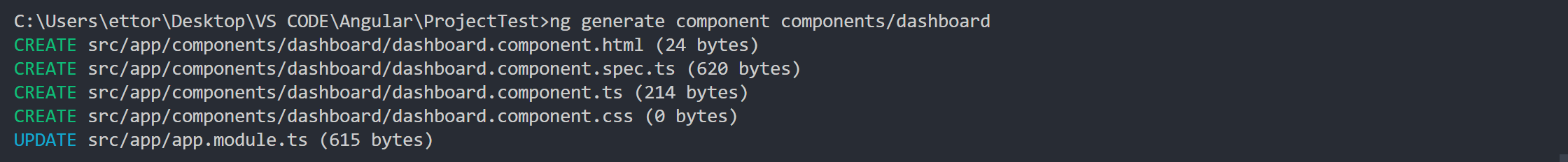
### 4.1.1 - Angular

Angular è un framework **open-source** sviluppato per la maggior parte da *Google*. Esso è implementato con *TypeScript*, utilizzando un approccio modulare che può **OOP** (Object Oriented Programming). La creazione di tali **componenti**, dunque, rende le Web Application dinamiche e scalabili.

Tra le caratteristiche principali di Angular troviamo la gestione del **routing**, l’iniezione delle **dipendenze**, la manipolazione avanzata del *DOM*, la gestione agile degli eventi e un sistema di **binding dei dati bidirezionale**.

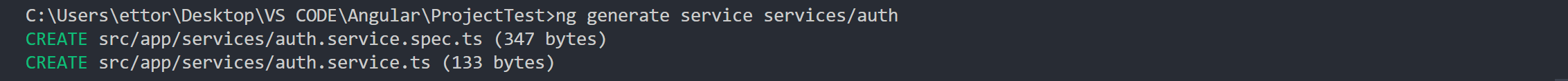
#### 4.1.1.1 - Componenti

Essi rappresentano dei blocchi che mostrano la suddivisione dell’applicazione. Sono composti da una struttura HTML, una logica di programmazione e i metadati per formare elementi modulari e riutilizzabili nelle interfacce utente delle applicazioni web. Questo approccio mira a migliorare la scalabilità del codice, utilizzando un approccio che agevola l’implementazione del sistema nel tempo.

*Creazione di un componente tramite Angular CLI. Se non esiste già, verrà generata automaticamente una cartella di nome “components” con dentro un’altra cartella “login” con dentro i file mostrati in figura.*

#### 4.1.1.2 - Services

Essi sono delle classi fondamentali per la condivisione di funzionalità tra i vari componenti all’interno dell’applicazione, inoltre, sono responsabili della gestione della comunicazione con il back-end. Difatti, grazie alla loro iniezione nei componenti, i servizi favoriscono la modularità e la scalabilità del codice, consentendo una gestione più efficiente e snodata delle diverse parti dell’applicazione

*Creazione di un service tramite Angular CLI. Se non esiste già, verrà generata automaticamente una cartella di nome “services” con dentro i file mostrati in figura**.*

#### 4.1.1.3 Routes

Essi sono strumenti fondamentali per la gestione della navigazione all’interno di una ***Single Page Application (SPA).*** La loro funzionalità è quella di definire percorsi specifici all'interno dell'applicazione e associare ognuno di essi a un componente specifico. Questi percorsi vengono poi riflessi nella barra degli Url del browser durante la navigazione dell’utente.

L’interazione tra percorsi e componenti semplifica la gestione della navigazione all’interno della *SPA*, poiché consiste di identificare in modo chiaro quale componente deve essere caricato per ciascun percorso, consentendo una navigazione fluida e intuitiva all'interno *dell'applicazione*.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteVengono definite due rotte: “registration” e “dashboard”, ognuna associata ad un componente specifico.

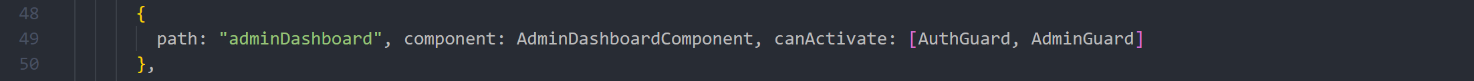
*4.1.1.4 Guards*

Le guards sono meccanismi di sicurezza essenziali in Angular che contribuiscono a proteggere le route dell'applicazione. Una guard viene utilizzata per decidere se un utente può accedere o meno a determinate parti dell'applicazione, basandosi su condizioni specifiche come l'autenticazione e i permessi dell'utente. Questo processo di verifica è cruciale per la gestione dell'accesso e per garantire che risorse sensibili siano esposte solo a utenti autorizzati. Le guards possono anche gestire reindirizzamenti automatici, ad esempio rimandando gli utenti non autenticati alla pagina di login.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Esempio di *auth.guard* che invoca la funzione nello *user.service,* la quale controlla se l’utente è autenticato, qualora non lo fosse, viene reindirizzato sulla pagina di login.

Esempio di applicazione di 2 guard sulla rotta adminDashboard, queste 2 Guards avranno il compito di verificare se l’utente è autenticato ed ha il ruolo “ADMIN”.

### 4.1.2 – TailwindCss

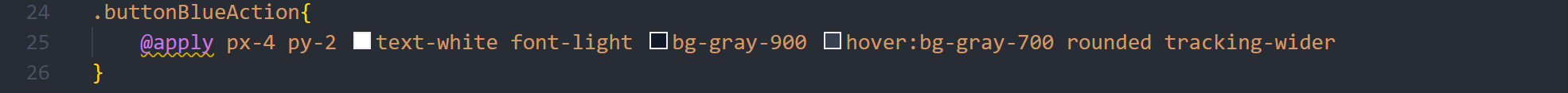
*TailwindCss* è un framework *CSS* progettato per agevolare lo sviluppo e la manutenzione del codice *CSS*. Esso si distingue dagli altri framework del settore per la sua fornitura di classi di utilità CSS **basate su singole proprietà**. Tra le sue caratteristiche troviamo:

**Leggibilità –** Le classi di utilità sono definite in modo intuitivo, aumentando quindi la comprensione del codice.

Esempio di utilizzo di una classe Tailwind in un elemento HTML.

**Basso livello di astrazione –** Tailwind si trova ad un livello molto basso di astrazione, consentendo un controllo più dettagliato e diretto degli stili CSS applicati.

**Elasticità –** È possibile personalizzare il set di classi nel file di configurazione per adattarlo a esigenze specifiche.



Esempio di creazione di una classe buttonBlueAction dove, tramite la keyword **@apply**, vengono applicate classi Tailwind già esistenti.

### 4.1.3 Leaflet

Libreria TypeScript open