Università degli Studi di Salerno
Corso di Ingegneria del Software

JapanWorld
System Design Document
Versione 1.1

Data: 17/08/2020

Coordinatore del progetto:

Nome	Matricola
Emanuele Patella	0512104730

Partecipanti:

Nome	Matricola
Emanuele Patella	0512104730

Scritto da:	Emanuele Patella

Revision History

Data	Versione	Descrizione	Autore
26/09/2020	1.0	SDD Versione 1	Emanuele Patella
27/09/2020	1.1	Revisione Documento	Emanuele Patella
30/09/2020	1.2	Completamento e revisione Subsystem decomposition	Emanuele Patella
1/10/2020	1.3	Hardware/Software Mapping	Emanuele Patella
2/10/2020	1.4	Revisione Hardware/Software Mapping	Emanuele Patella
4/10/2020	1.5	Persistent data Management	Emanuele Patella

Indice

1. Introduzione

1.1. Scopo del sistema

Lo scopo del sistema è quello di istanziare online una piattaforma che permetta la compravendita di prodotti audio-visivi a tema Anime e Manga.

I servizi forniti dall'infrastruttura online saranno quelli tipici dell'e-commerce online, quindi il rilascio dovrà avvenire fornendo tutte quelle che sono le informazioni necessarie alla sicurezza dei dati trattati.

1.2. Obiettivi di Design

Gli obiettivi di design sono i seguenti:

- Creare un sito la cui interfaccia HTML sia responsiva ai diversi schermi dei device con i quali verrà visualizzata;
- Creare un database programmato in SQL da istanziare su di un server per poter tenere traccia dei dati necessari per il buon funzionamento del nostro sito web;
- Rendere impossibile per gli utenti accedere in maniera diretta ai file di back-hand, per loro dovrà essere possibile lavorare solo tramite l'interfaccia creata appositamente per lo scopo;
- Implementare delle Query studiate in modo tale da evitare che un input non opportuno mandi in blocco il sistema. Gli unici problemi che ci si presuppone di non poter gestire preventivamente sono quelli che avvengono in maniera imprevista, quali crash del sistema causa crollo delle linee o altri problemi dovuti alle infrastrutture e non al sistema in sé.

1.2.1. Criteri e trade off

Il sistema verrà sviluppato sfruttando il linguaggio HTML perché il cliente e gli sviluppatori si sono accordati su questo linguaggio, in quanto gli sviluppatori sono fiduciosi nella loro conoscenza del linguaggio, mentre il cliente non ha fatto particolari richieste in merito.

1.2.2. Criteri delle performance

Troughtput: Il sistema dovrà avere dei tempi di reazione il quanto più possibile brevi, poiché, trattandosi di un e-commerce, rendere lungo e tedioso il caricamento tra una schermata e l'altra inficerebbe negativamente sulla usufruibilità del sito.
Più nello specifico:

- Dal momento della digitazione dell'url del sito, l'interfaccia della home dovrà essere completamente carica e funzionante nel giro di 2 secondi al massimo;
- Dal momento del click dell'utente su una delle categorie o l'inserimento di un filtro di ricerca nella apposita barra al momento del caricamento dei risultati della ricerca, dovranno passare 3 o 4 secondi al massimo;
- Dal momento in cui l'utente clicca il bottone per aggiungere un prodotto al carrello, all'effettiva aggiunta del prodotto, dovranno trascorrere 0,5 secondi;
- Dal momento in cui l'utente clicca sul pulsante per aprire il proprio carrello, il tempo stimato per caricare completamente il carrello con i prodotti e potervi operare è di 1 secondo;
- La rimozione di un prodotto dal carrello, proprio come l'aggiunta di un prodotto ad esso, non dovrà richiedere più di mezzo secondo;
- Dal momento in cui l'utente clicca sul bottone per avviare le pratiche di pagamento la relativa interfaccia deve essere completamente carica e operativa nell'arco di 1 secondo;
- Nella versione definitiva del sito, la sezione dedicata alla transazione vera e propria sarà affidata ai singoli siti delle banche stesse, quindi, non ci arroghiamo il diritto di definire i tempi di reazione di periferiche di terze parti che utilizzeremo;
- Quando l'utente clicca sul pulsante login, l'interfaccia deve essere completamente carica e funzionante nell'arco di 1 secondo;
- Quando l'utente clicca sul pulsante login dopo aver completato i campi, la verifica e l'eventuale accesso alla pagina personale deve essere completata entro 2 secondi;
- Uscire dal profilo personale dovrebbe richiedere 0,5 secondi dal click del pulsante.

1.2.3. Costi di sviluppo

Lo sviluppo del sito richiederà, all'incirca, 7 mesi di lavoro da parte del team di viluppo.

Più nello specifico, serviranno 2 mesi per la progettazione e elaborazione del sistema, altri 2 mesi per l'effettivo sviluppo software, e i restanti 3 saranno necessari per lo sviluppo di un piano di testing appropriato e la sua messa in opera.

1.2.4. Affidabilità del sistema

Il sistema è sato progettato per incorrere il meno possibile in errori che possano mandarlo completamente down qualora occorressero, infatti tutte le operazioni eseguibili da utente e proprietario del sito sono estremamente guidate, impedendo il grosso dei problemi proprio a livello di usabilità della piattaforma.

Gli utenti che cercano di registrarsi vengono avvisati attraverso messaggi forniti dal sistema stesso qualora i loro dati non fossero corretti secondo la formattazione del sito.

Unici problemi che possono occorrere in maniera improvvisa e rendere, nella quasi totalità, il sistema inutilizzabile, sono quelli lato server, qualora l'accesso ai dati non fosse possibile per via di problemi di comunicazione tra client e server il sistema lo notificherà con un messaggio apposito e avviserà il proprietario che è necessaria una riparazione da parte del personale preposto.

1.2.5. Trade-Off

Trade-off	
Manutenibilità vs Performance	Il reenginering del sistema da priorità alle performance dello stesso, in quanto dei tempi di risposta brevi sono fondamentali per la fruibilità di un e-commerce. Questa nostra scelta fa si che il sistema abbia degli ottimi tempi di risposta, ma può andare ad inficiare sulla leggibilità del codice per eventuali modifiche future.
Performance vs Sicurezza	Il reenginering del sistema da priorità alla sicurezza, con l'utilizzo di algoritmi di cifratura. Trattandosi di una piattaforma online che andrà a trattare dati importanti, quali

	nominativi e carte di credito, è stato kobbligatorio concentrarci su questo aspetto del software.
Manutenibilità vs Sicurezza	Il reenginering del sistema da priorità alla sicurezza per lo stesso motivo di cui sopra. Il sistema fonderà gran parte della sua vita sul fato che sia sicuro, quindi questo punto deve essere sviluppato al meglio.
Usabilità vs Sviluppo rapido	Il reenginering del sistema da priorità all'usabilità del sistema, in quanto, questo tipo di sistema(e-commerce), si propone ad un pubblico estremamente vasto, che include individui delle più disparate fasce d'età, e con non necessariamente competenze informatiche, quindi l'usabilità è un qualcosa su cui il sistema deve spingere per far sì che il sito sia accattivante una volta online.
Robustezza vs Portabilità	Il reenginering del sistema da priorità alla robustezza del sistema. Questa scelta è stata fatta, fondamentalmente, su richiesta dei programmatori, in quanto, per favorire la sicurezza, un sistema robusto è molto più comodo. Il sistema sarà compatibile con i sistemi windows.

1.3. Definizioni, acronimi e abbreviazioni

All'interno del codice e dei progetti non sono presenti acronimi o abbreviazioni, l'unico tipo di definizione usata all'interno dei documenti sono i termini Anime e Manga, i quali fanno, rispettivamente, riferimento ai prodotti di Animazione e ai fumetti Giapponesi.

1.4. Riferimenti

Il sistema che andremo ad implementare, come già richiesto dal cliente nel proble statement, Dovrà avere una interfaccia estremamente intuitiva, ma accattivante, a questo scopo, all'interno del RAD, sono stati inseriti i mock-up per dare al cliente un'idea di come sarà l'interfaccia finale.

È stata anche assegnata una palette di colori da parte del committente e della quale si terrà conto nella fase di implementazione dell'interfaccia grafica del sito.

1.5. Overview

Il nostro progetto dovrà porsi in un ambiente online che comprende una fascia estremamente vasta di utenti, quindi il progetto richiederà uno studio molto approfondito per rendere la piattaforma si accattivante, ma anche intuitiva e facile da utilizzare.

2. Architettura corrente del software

Con questo progetto non stiamo andando a sostituire una piattaforma già esistente ma ne stiamo creando una nuova per un negozio non ancora entrato nel mondo degli e-commerce.

Le infrastrutture simili già esistenti fanno molto affidamento su di una base skill elevata degli utenti, in quanto l'e-commerce di un negozio, di base, assomiglia molto a quello degli altri, nelle sue funzioni basilari, e poi vengono aggiunte funzionalità specifiche in base al tipo di prodotti che si va a vendere, o del tipo di utenza che, principalmente, ci si vuole accaparrare.

Nel nostro caso l'idea di fondo è prendere queste funzionalità specifiche e innovative che vengono a formarsi in ogni piattaforma, e rimuoverle dall'equazione, creando una piattaforma più snella e semplice da utilizzare.

Le uniche assunzioni che andremo a fare, quindi, sono:

- Gli utenti sanno cosa sia una iscrizione a una piattaforma online;
- Gli utenti conoscono la terminologia di base delle piattaforme online(Login, ecc…);
- Gli utenti conoscono la tipologia di prodotti di cui si occupa il rivenditore, quindi non rimarranno straniti da diciture, quali Anime, Manga e simili.

3. Architettura proposta del sistema

3.1. Overview

Il software ha una struttura ad albero radicato, con la sua radice nella Home Page, da questa schermata, sarà possibile avviare tutte le attività, tuttavia sono presenti anche dei collegamenti tra punti intermedi delle varie attività grazie alla barra di navigazione presente in alto nelle pagine, tuttavia no sarà possibile spostarsi da una attività all'atra in tutti i punti del software, poiché alcune operazioni richiedono di essere completate in serie, altrimenti i dati necessari verrebbero compromessi, rendendo impossibile il completamento.
L'architettura che più si confà alla creazione di una piattaforma di questo genere, è l'architettura MVC, quindi opteremo per quest'ultima nella realizzazione del nostro e-commerce.

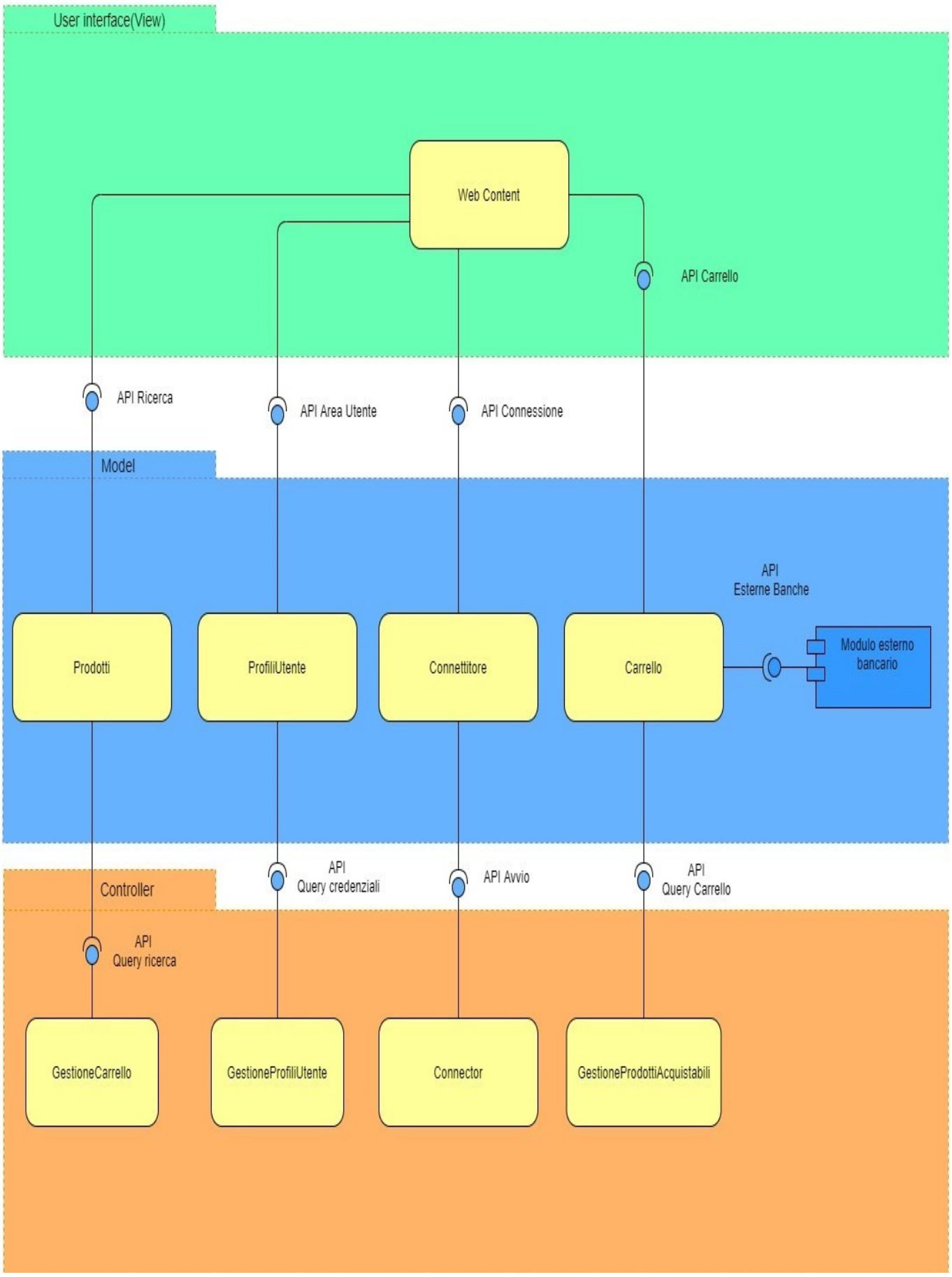
3.2. Decomposizione dei sottosistemi

User Interface (View)	***Web Content*** Questo package sarà quello che fornirà tutte le funzioni di tipo grafico, necessarie alla visualizzazione dei dati da parte dell'utente. Le API che fornirà sono principalmente 3: • API Ricerca: Relative alla gestione e alla visualizzazione dei dati relativi alle ricerche di prodotti effettuate dall'utente; • API Area Utente: Relative alla visualizzazione della interfaccia utente all'interno dell'area privata e alla ricezione in ingresso dei dati di Login e Registrazione; • API Carrello: Relative alla visualizzazione e all'interazione con prodotti aggiunti al carrello da parte dell'utente. SI occuperà anche di Gestire l'interfaccia grafica relativa ai pagamenti. • API Connessione: Necessarie per richiamare, all'avvio del sito, la componente del model che, a sua volta, avvierà la connessione con il database.
Model	***Model*** Questo package, omonimo della componente che rappresenta, si incentrerà sul far funzionare tutte le interazioni presenti

	all'interno dell'interfaccia e di fare da tramite tra la View e il controller. Le API fornite da questo package saranno 4, di cui una esterna al progetto: • API Query Ricerca: Saranno i metodi necessari a processare i dati in input nell'interfaccia e passarli in maniera appropriata alle rispettive componenti del controller; • API Query Credenziali: Saranno tutti quei metodi relativi alla presa in input dei dati dell'utente e alla formattazione per la verifica dei suddetti dati, necessari per accedere all'area riservata dell'utente stesso; • API Query Carrello: Saranno tutti quei metodi che i occupano di gestire le funzionalità implementate nel carrello, formattando adeguatamente i dati contenuti nella session per poi permettere al controller correlato di utilizzarli per visualizzare, nella loro interezza, i prodotti presenti nel carrello ed interagire con loro; • API Avvio: Necessarie a stabilire la connessione con il database all'avvio della piattaforma. • API Esterne Banche: Componente di API esterna al sito, che verrà sfruttata per effettuare la verifica dei dati inseriti per completare la transazione durante l'acquisto dei prodotti.
	Modulo esterno bancario Necessario all'interno del grafico per rendere evidente a tutti che il controllo delle carte, e dei dati ad esse connessi, non è di competenza della piatatforma in sviluppo.
Controller	***Controller*** Package omonimo della componente che rappresenta, si occuperà di tutte le interazioni necessarie per permettere al database di comunicare col sito e di aggiornarlo in tempo reale, ogni volta che una transazione viene conclusa.

Grafico a pagina seguente:↓

Subsystem decomposition

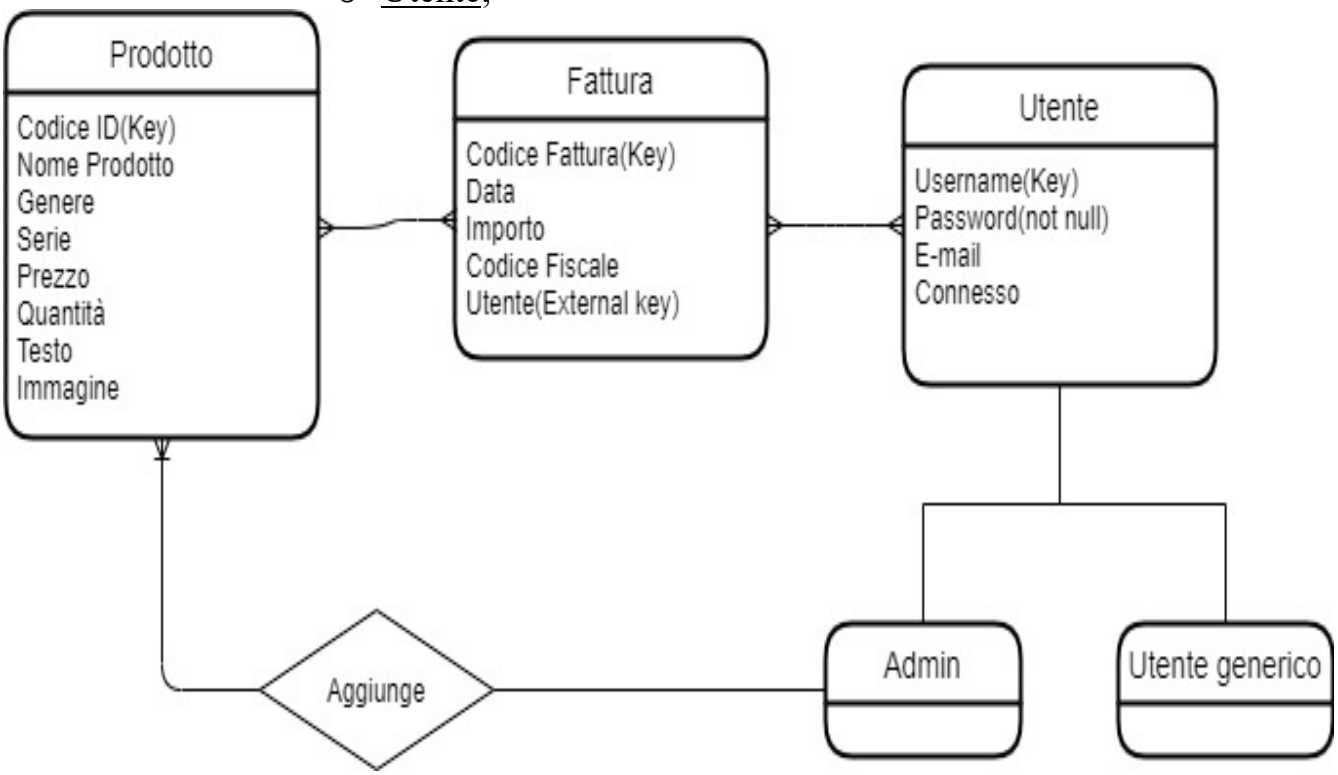


3.3. Persistenza dei dati

I dati da rendere persistenti all'interno della nostra piattaforma sono molti, quindi opteremo per un database, il quale favorirà anche la portabilità e la scalabilità della struttura stessa, qualora il negozio venisse ampliato o decidesse di diventare una catena.
Il database, come richiesto da contratto, sarà sviluppato in SQL e sarà composto da 3 tabelle:

- Utente: che conterrà i dati di tutti gli utenti registrati sul sito, e avrà 5 colonne:
    - Username;
    - Password;
    - E-mail;
    - Admin;(Variabile booleana che indica se l'utente è, oppure no, l'admin del sito[Il profilo admin di default inserito nel database userà il classico "admin", "admin", "admin@hotmail.it" come username, password e e-mail])
    - Connesso.(Variabile necessaria ad alleggerire il numero di dati che devono essere mantenuti in sessione durante la navigazione e che serve a verificare che l'utente sia o meno loggato col proprio profilo)
- Prodotto: che conterrà tutti i dati relativi ai prodotti in vendita in negozio, avrà 8 colonne:
    - Codice ID: codice univoco del prodotto(Key);
    - Nome Prodotto;
    - Genere: Necessario per distinguere i prodotti cartacei da quelli audiovisivi;
    - Serie: Indica la serie di appartenenza, parametro necessario per le ricerche;
    - Prezzo;
    - Quantità: Indica le unità di prodotto presenti nel negozio;
    - Testo: Piccola descrizione del prodotto, facoltativa;
    - Immagine: Indica la cartella e il singolo file dell'immagine da visualizzare per il prodotto.
- Fattura: Tabella che conterrà tutte le transazioni effettuate dai singoli utenti e i prodotti da loro acquistati. Indicata come relazione acquista, viene reiterata come fattura nella creazione del database per coerenza con la componente corrispondente nel codice. Sarà composta da 5 colonne:

- Data: indica la data di fatturazione, né più, né meno;
- Codice Fattura: Identificativo univoco della fattura, che corrisponde al codice del prodotto acquistato;
- Codice Fiscale: necessario per la fatturazione;
- Importo;
- Utente;



3.4. Controlli d'accesso e sicurezza dei dati

	User Interface	**Gestione Profili**	**Ricerca Prodotti**	**Carrello**	**Gestione persistenza dati**
Utente	Ha accesso				
Sistema		Ha accesso	Ha accesso	Ha accesso	Ha accesso

Nel sistema da me sviluppato l'utente sarà impossibilitato ad accedere direttamente ai sottosistemi che si occupano delle operazioni compiute dal sito, tuttavia la User Interface farà da tramite per l'utente, forzandolo ad inserire i dati

con una corretta formattazione, impedendo quindi molti dei problemi che il sistema potrebbe avere a causa della formattazione non consona dei dati.

Nella versione definitiva del sistema andrà anche implementato un sistema di cifratura dei dati, per far si che i profili dei clienti non vengano memorizzati in chiaro(per ora, la fase di sviluppo avverrà con dati memorizzati in chiaro per facilitare la risoluzione di eventuali errori di implementazione).

La gestione delle key sarà completamente delegata al sistema, in quanto l'utente inserirà la propria mail come chiave, ma sarà il sistema ad avvisarlo se quest'ultima è valida oppure è già stata inserita all'interno del database, impedendo, di conseguenza, la creazione dell'account.

3.5. Boundary condition

Le richieste che il sistema dovrà processare saranno gestite nel modo seguente:

1. L'interfaccia grafica permetterà l'inserimento dei dati da parte dell'utente in odo che rispettino la formattazione richiesta dal sistema stesso;
2. A questo punto, nel momento in cui i dati verranno confermati da parte dell'utente attraverso il comando apposito(Click sul bottone "Conferma"), la query verrà presa in carico dal sistema che la formatterà e la invierà a un sottosistema apposito che si occuperà di eseguire la query sul database e restituire, con la formattazione corretta, i dati all'interfaccia grafica, di modo che vengano poi visualizzati da parte dell'utente.
3. Il processo verrà ripetuto finché l'utente non avrà raggiunto il completamento dell'operazione da lui desiderata.

Esempio: ***L'utente vuole acquistare un prodotto***

Il processo di inserimento e conferma dei dati andrà ripetuto all'incirca 2 volte per effettuare completamente l'operazione:

1. La prima volta dovrà richiedere al sistema di procedere all'acquisto;
    a. A questo punto il sistema gli chiederà di inserire i dati per il pagamento;
2. L'utente dovrà compilare i campi e premere su conferma per permettere all'ordine di essere eseguito;

a. Il sistema prenderà in carico i dati e ne verificherà la validità attraverso le periferiche fornite dall'ente preposto al pagamento;
b. Il sistema, se i dati risultassero validi per la transazione, procederebbe alla fase successiva, completando la transazione e fornendo lo scontrino digitale della transazione all'utente;

3. Presa visione dello scontrino, l'operazione giungerà al termine anche dalla parte dell'utente, permettendogli di tornare a navigare il sito in libertà o di chiudere la piattaforma.

La sincronizzazione tra sottosistemi sarà garantita dalla sequenzialità delle operazioni, le quali permetteranno ai singoli sottosistemi di fermarsi e attendere il completamento delle operazioni necessarie da parte degli altri sottosistemi.

Il caso di crash va suddiviso, a sua volta, nei due tipi di crash che si possono verificare, cioè:

- Crash dovuto a problemi alla rete internet(Gestito dal sistema);
- Crash dovuto all'interruzione fisica del servizio, per esempio, malfunzionamenti dei server sui quali è istanziato il nostro software(Gestito parzialmente).

ID	CRASH Internet
Participating actors	Clienti
Entry condition	Gli utenti stanno utilizzando il sito web
Flow Of events	**Utente** **Sistema** 1. **L'utente sta navigando il sito quando la connessione viene meno.** 2. Il sistema non riscontra l'effettiva assenza dell'utente ma mantiene

	salvate nella session i dati relativi all'ultima operazione svolta dall'utente. **3. L'utente ricarica la pagina, una volta ripristinata la propria connessione.** 4. Il sistema ricarica gli ultimi dati che ha salvato nella session, ripotando l'utente all'ultimo passaggio utile.
Exit condition	La connessione viene ripristinata
Exeption condition	
Quality requirement	

ID	CRASH Hardware
Participating actors	Clienti, Operatore preposto
Entry condition	Gli utenti stanno utilizzando il sito web
Flow Of events	**Utente** **Sistema** **Operatore** **1. L'utente sta navigando il sito quando i server su cui è istanziato il sito web vengono meno.** 2. Il sistema lavora su di una copia dei dati del database, e aggiorna l'originale solo ad operazione compiuta, quindi, al verificarsi ti questo problema, i dati originali non vengono intaccati. **3. L'utente deve attendere la risoluzione del problema da parte di**

	un operatore preposto. 4. L'operatore viene inviato a riavviare i server. 5. Il sistema ripristina gli ultimi dati presenti nella versione originale del database, quindi, solo le ultime operazioni risulteranno perse. **6. L'utente può riprendere adesso a utilizzare il sito.**
Exit condition	I server sono di nuovo online
Exeption condition	
Quality requirement	

4. Subsystem services

I servizi sui quali il software si andrà a poggiare sono principalmente 2:

- API Ricerca: Da *Web Content* a *Prodotti*. Sono l'insieme di API che si occuperanno di prendere in input i dati che l'utente inserirà tramite l'interfaccia del sito web, quindi, o dei filtri scelti tra quelli disponibili, oppure un filtro personalizzato tramite l'inserimento di testo all'interno dell'apposita barra. Queste specifiche API formatteranno i dati in ingresso per permettere alla componente del Model *Prodotti* di inserirli all'interno delle varie query per poi poterle passare ai controller che se ne occuperanno.
- API Area Utente: Da *Web Content* a *ProfiliUtente*. Queste API sono preposte alla gestione dell'input dei dati da parte dell'utente, username e password nello specifico, e alla verifica della validità dei suddetti. I dati vengono passati come parametri e le API restituiscono un valore vero qualora le credenziali corrispondano e uno falso qualora non corrispondano. C'è da specificare che, nel caso sia la funzione di Iscrizione e non quella di Login a essere avviata, il valore di ritorno è invertito, poiché la registrazione può avvenire solo se l'utente non è duplicato e quindi il valore non è già presente in tabella. Una volta effettuata questa specifica verifica, il nuovo utente viene aggiunto al database.
- API Connessione: Da Web *Content* a *Connettitore*. Questa API viene eseguita all'avvio del sistema e ha il compito, insieme alla componente *Connector* di avviare la connessione con il database, permettendo, di fatto, a tutto il sistema di funzionare. C'è da specificare che queste API e la componente *Connettitore* sono solo un ponte per arrivare al controller che effettuerà l'operazione vera e propria, ma era necessario mantenere la suddivisione del modello MVC.
- API Carrello: Da *Web Content* a *Carrello*. Queste API si occupano di gestire tutte le interazioni tra l'utente e il suo carrello. Queste API verranno richiamate ogni qual volta l'utente aprirà il suo carrello e si occuperanno anche di gestire la rimozione dei prodotti dal carrello e la loro ri-aggiunta nell'elenco dei prodotti disponibili per l'acquisto.
- API Esterne Banche: Da *Carrello* a *Modulo esterno Bancario*. Queste API non verranno scritte dallo stesso staff che si occuperà della

realizzazione del sito, bensì, verranno sfruttare e integrate quelle già esistenti, di proprietà delle banche, e verranno implementate con la tecnica Black Box.

- API Query Ricerca: Da *Prodotti* a *GestioneCarrello*. Queste API avranno un ruolo importante all'interno del nostro sito web. Effettueranno tutte le query necessarie a effettuare la ricerca dei prodotti all'interno del database, restituendo una stringa di dati da formattare per renderli visualizzabili. Gestirà le modifiche alle quantità all'interno del database dei prodotti disponibili e aggiungerà i prodotti al carrello quando l'utente premerà l'apposito tasto.
- API Query Credenziali: Da *ProfiliUtente* a *GestioneProfiliUtente*. Queste API si occupano delle query sia di Login che di Registrazione, poiché differiscono solo per una credenziale in più presente nel form di Registrazione. Le API effettuano la query al database usando le credenziali appena inserite e restituiscono un valore vero se l'utente è già presente nel database, falso altrimenti, per quanto riguarda il Login. Restituiscono un valore vero se l'utente non è presente nel database e falso altrimenti, se si sta effettuando la Registrazione.
- API Avvio: Da *Connettitore* a *Connector*. Questa API viene eseguita all'avvio della piattaforma per permettere la connessione al database. Viene avviata tramite il *Connettitore* che fa da ponte nel modello MVC, al fine di non perdere l'architettura del nostro sito. Le API che verranno usate durante lo sviluppo, più nello specifico, sono quelle di MySQL(servlet-api), queste API possono essere soggette a cambiamenti qualora si volesse cambiare il linguaggio del database.
- API Query Carrello: Da *Carrello* a *GestioneProdottiAcquistabili*. Queste API si occuperanno di gestire la visualizzazione dei prodotti nel carrello attraverso query apposite. Si occuperanno anche di avviare le pratiche di pagamento per completare l'acquisto e, soprattutto, gestiranno la rimozione dei prodotti dall'interno del carrello e la loro ri-aggiunta all'interno del catalogo dei prodotti disponibili.

Non sarà necessario implementare API per permettere la comunicazione tra View e Model, in quando quest'ultima è garantita già in fase di programmazione dal linguaggio di sviluppo Java.