UNIVERSITÀ DEGLI STUDI DI SALERNO
Corso di Ingegneria Software

FATTORIA ROSABELLA
SYSTEM DESIGN DOCUMENT
VERSIONE 1.1

FATTORIA ROSABELLA
1992

17/12/2020

Progetto: FATTORIA ROSABELLA	Versione: 1.1
Documento: System Design Document	Data: 17/12/2020

Coordinatore del progetto:

Nome	**Matricola**
Mignoli Pasquale Antonio	0512105826

Partecipanti:

Nome	**Matricola**
Mignoli Pasquale Antonio	0512105826
Pastore Carmine	0512106058
Laurino Nicola	0512105970

Scritto da:	Mignoli, Laurino, Pastore

Revision History

Data	**Versione**	**Descrizione**	**Autore**
14/12/2020	1.0	Inizio stesura S.S.D.	Mignoli Pasquale
17/12/2020	1.1	Aggiunta del Data Persistent Manager, sicurezza, controllo del flusso e condizioni limite.	Mignoli Pasquale Carmine Pastore Laurino Nicola

Sommario

Introduzione

1.1 Scopo del sistema

Lo scopo del sito web che sviluppiamo per *Fattoria Rosabella* è quello di fornire una piattaforma web per la prenotazione delle attività proposte al pubblico. In particolare, il sito è accessibile a tutti gli utenti attraverso un qualsiasi browser, dunque non richiede istallazione fisica. Inoltre, è accessibile attraverso qualsiasi dispositivo, che sia un computer, uno smartphone o un tablet.

1.2 Design goals & Trade-offs

Di seguito vengono riportati gli obiettivi del sistema e le relative priorità, per chiarezza a obiettivi con priorità alta viene associato un valore numerico più basso. Ciascun obiettivo fa riferimento ad un particolare requisito non funzionale, dunque vengono riportati anche questi ultimi.

Priorità	ID	Descrizione	Categoria	Origine
1	DG1	**Sicurezza**. Il sistema prevede l'inserimento di dati sensibili da parte dell'utente e quindi si preoccupa di utilizzare tecnologie appropriate al fine di garantire la riservatezza dei dati.	Dependability	RNF8
1	DG2	**Usabilità**. Il sistema è web based, dunque prevede interfacce grafiche per l'interazione dell'utente con il sito. Dunque, risulta strategico l'uso di interfacce semplici ed intuitive. Inoltre, il sito prevede l'implementazione di moduli response per garantire l'usabilità da qualsiasi dispositivo (desktop, tablet e mobile).	End user	RNF1
2	DG3	**Costi di sviluppo**. Lo sviluppo della webapp prevederà costi ridotti, in termini di risorse umane e tecnologie usate.	Costi	RNF5
2	DG4	**Modificabilità**. Il sistema deve essere facilmente modificabile al fine di garantire semplicità nella modifica di nuove funzionalità in futuro.	Manutenibilità	RNF4
2	DG5	**Estensibilità**. Il sistema deve garantire una facile aggiunta di nuove funzionalità in futuro, al fine di permettere l'introduzione di nuove funzioni.	Manutenibilità	RNF4
3	DG6	**Tempi di risposta**. Il sistema deve fornire risposte alle interazioni degli utenti in un tempo massimo di tre secondi.	Performance	RNF3
3	DG7	**Robustezza**. Il sistema prevede diversi form attraverso il quale l'utente può immettere dati. La webapp deve allora garantire una buona tollerabilità ad input errati immessi dall'utente, segnalarli e indicare come correggerli.	Dependability	RNF2
3	DG8	**Affidabilità**. Il sistema deve garantire consistenza con i dati del database, prevedendo le modifiche.	Dependability	RNF2

1.2 Definizioni, acronimi e abbreviazioni

- **Fattoria Rosabella**
  Rappresenta il nome dell'azienda per il quale il sistema viene sviluppato.
- **Gestore**
  Rappresenta il soggetto fisico che si occupa di aggiornare il sito.

1.3 Riferimenti

- Problem Statement (PS)
- Requirements Analysis (RA)

1.0 Sistema corrente

La Fattoria Rosabella non possiede alcun sito attualmente, dunque il sistema che si va a sviluppare non va a sostituire alcun sistema esistente.

Prendiamo allora come piattaforma di considerazione parcodelgrassano.it, un'applicazione web di un bioparco che permette di prenotare un tavolo per il pic nic online. Esso è raggiungibile attraverso un qualsiasi browser e da qualsiasi dispositivo. È presente una sezione per la registrazione attraverso il quale un utente può creare un account e gestire in comodità, attraverso il portale, le sue registrazioni.

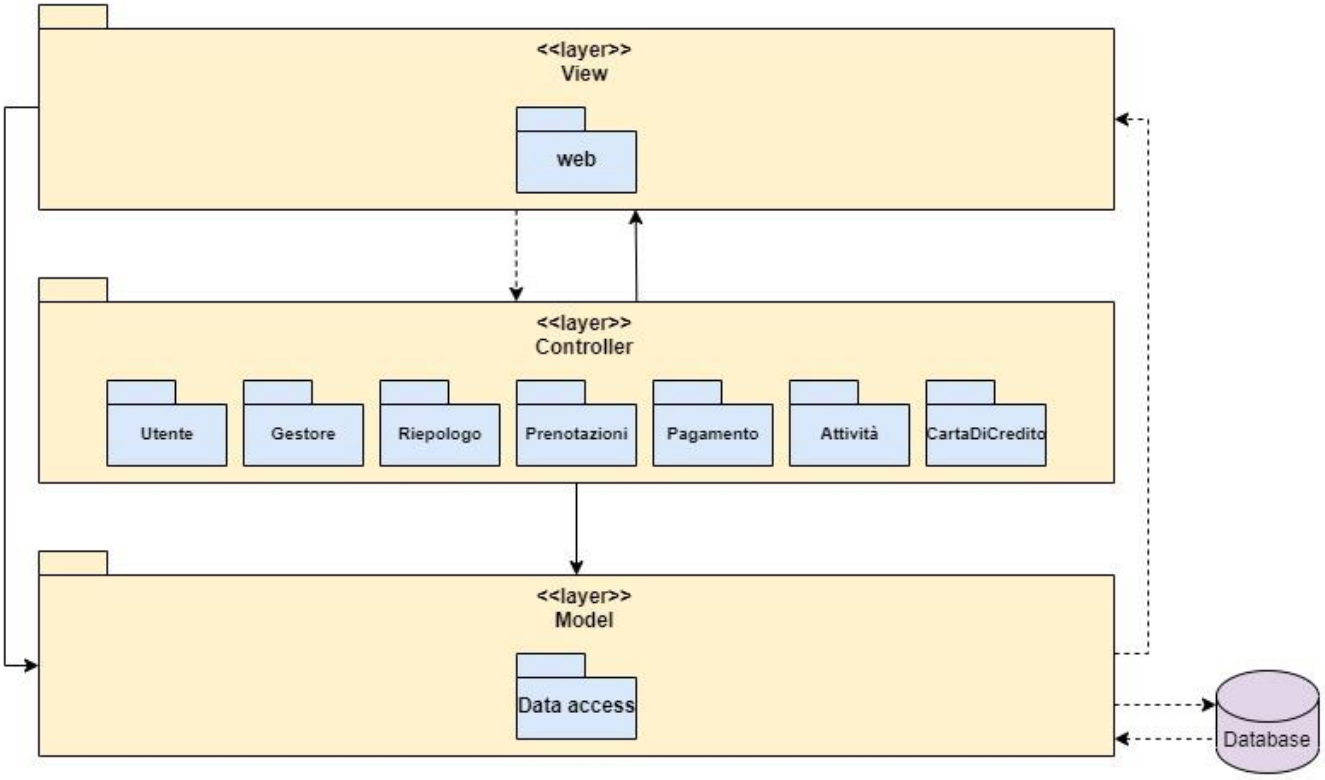
Attraverso un'analisi del sito sopracitato, è facile convincersi come venga adoperato un database per memorizzare le informazioni degli utenti, quali i dati personali e le carte di credito associate.

2.0 Sistema proposto

3.1 Overview

Il sito che andremo a sviluppare per Fattoria Rosabella sarà una web app, con un'interfaccia utente grafica e di semplice navigazione. Le autenticazioni al sito verranno gestite attraverso accurati controlli, garantendo così la sicurezza e la riservatezza degli account registrati, e i dati verranno salvati in modo permanente sfruttando un database relazione.

3.2.1 Decomposizione in sottosistemi



Il sistema viene suddiviso in tre grandi livelli, secondo un'architettura MVC. Dunque, vengono individuati le seguenti macrocategorie.

1. Livello di *view*.
   In questa categoria vengono definite tutte le pagine client, contenenti la parti grafica del sito, quindi interfacce grafiche, form, cards, informazioni sul bioparco e i contatti. Attraverso queste sezioni l'utente riesce a navigare all'interno del sito e ad interagire con l'intero sistema: fornisce i dati per la registrazione (pagina di *registrazione*), email e password per l'autenticazione (pagina di *login*), selezionare una data per visualizzare le attività disponibili (pagina di *prenotazioni*), registrare una nuova carta di credito e effettuare un pagamento (pagina di *pagamento*) e così via attraverso tutte le funzionalità della piattaforma.

2. Livello di *controller*.
   Questa categoria individua il livello di business della webapp, dunque tutti i moduli incaricati di gestire la logica dell'applicazione. In particolare, possiamo individuare:
   a. Server page dell'*utente*, incaricata della gestione dell'utente generico;
   b. Server page del *gestore*, incaricata della gestione dell'utente gestore il quale può aggiungere, modificare e/o rimuovere un'attività;

c. Server page di *riepilogo*, incaricata di controllare l'ordine che sta effettuando il cliente e indirizzarlo alla pagina di pagamento;
d. Server page della *prenotazione*, consente di visionare, modificare o annullare una o più prenotazioni;
e. Server page di *pagamento*, incaricata di gestire la parte finale di una prenotazione, ovvero il pagamento;
f. Server page delle *attività*, incaricata di visualizzare tutte le attività offerte dal bioparco;
g. Server page delle *carte di credito*, consento di gestire le proprie carte di credito, dunque aggiungerne di nuove, modificare quelle già presenti o eliminare quelle non più utilizzate.

3. Livello di *model*.
   Questa è la categoria incaricata di gestire la connessione tra la piattaforma webapp e il database, permettendo a due tecnologie diverse di comunicare tra loro fornendo così la possibilità di poter salvare i dati in modo permanente e di recuperarli quando richiesto.

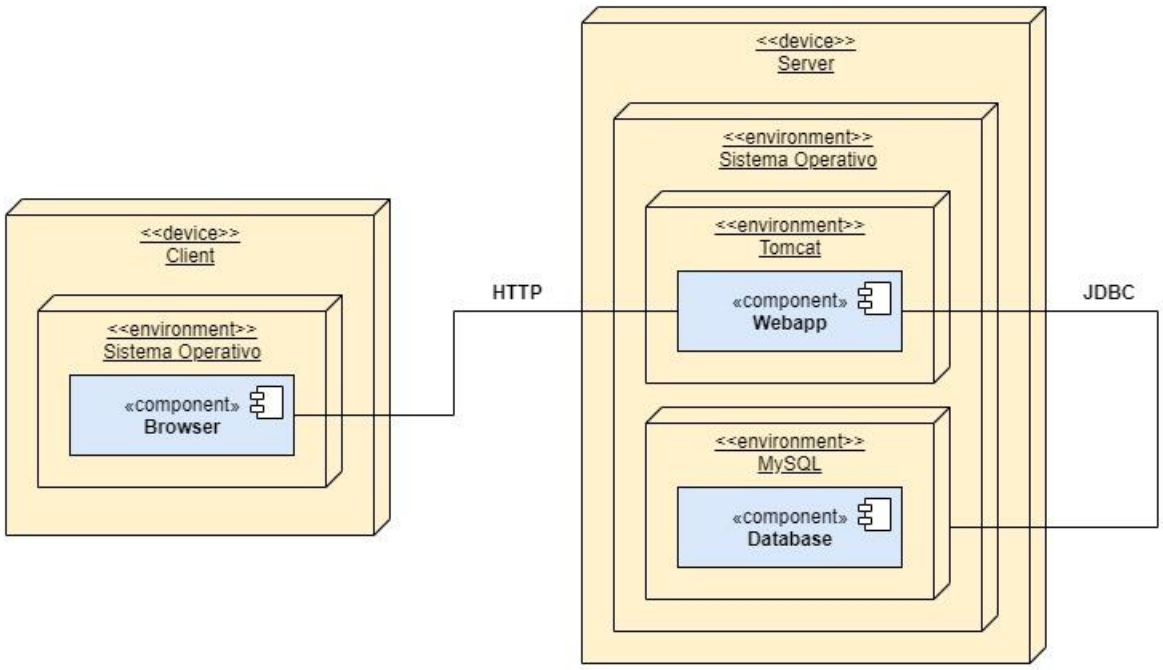
Come già detto, la suddivisione adottata è un pattern MVC, acronimo di *Model – View – Controller*, in quanto permette di separare in modo netto il livello che si occupa della logica di business da quello che si occupa della permanenza dei dati e da quello che ci occupa di gestire le interfacce utente. In questo modo, i tre livelli riescono a comunicare tra di loro senza conoscere come essi sono implementanti, facilitando molto l'aggiornamento della webapp in futuro.

3.2.2 Diagramma di deployment

La webapp per Fattoria Rosabella verrà sviluppata come un'applicazione distribuita, la logica di business verrà implementata unicamente in linguaggio Java mentre la persistenza dei dati verrà realizzata attraverso il tool MySQL. Per queste ragioni, viene richiesto un server che supporti le tecnologie appena citate per garantire un corretto funzionamento.

Non vengono previsti grossi carichi di utenti contemporaneamente connessi, dunque la webapp verrà sviluppata con un approccio client - server e le componenti verranno installate sulla stessa macchina.

Il sito web sarà accessibile attraverso un qualsiasi browser con un qualsiasi device che supporta quest'ultima applicazione.



3.3 Mapping di Hardware/Software

La webapp che andremo a sviluppare per Fattoria Rosabella sarà composta da due componenti principali:

1. La webapp vera e propria, che integra al suo interno il livello di *controller*, dunque la logica di business dell'applicazione, il livello di *view*, che consente di organizzare i dati e di visualizzarli attraverso una interfaccia grafica intuitiva per l'utente e i diversi *beans*, che individuano gli oggetti che fanno parte dell'applicazione.
2. Il database relazionale, il quale rappresenta il livello di *model* e si occupa della persistenza dei dati.

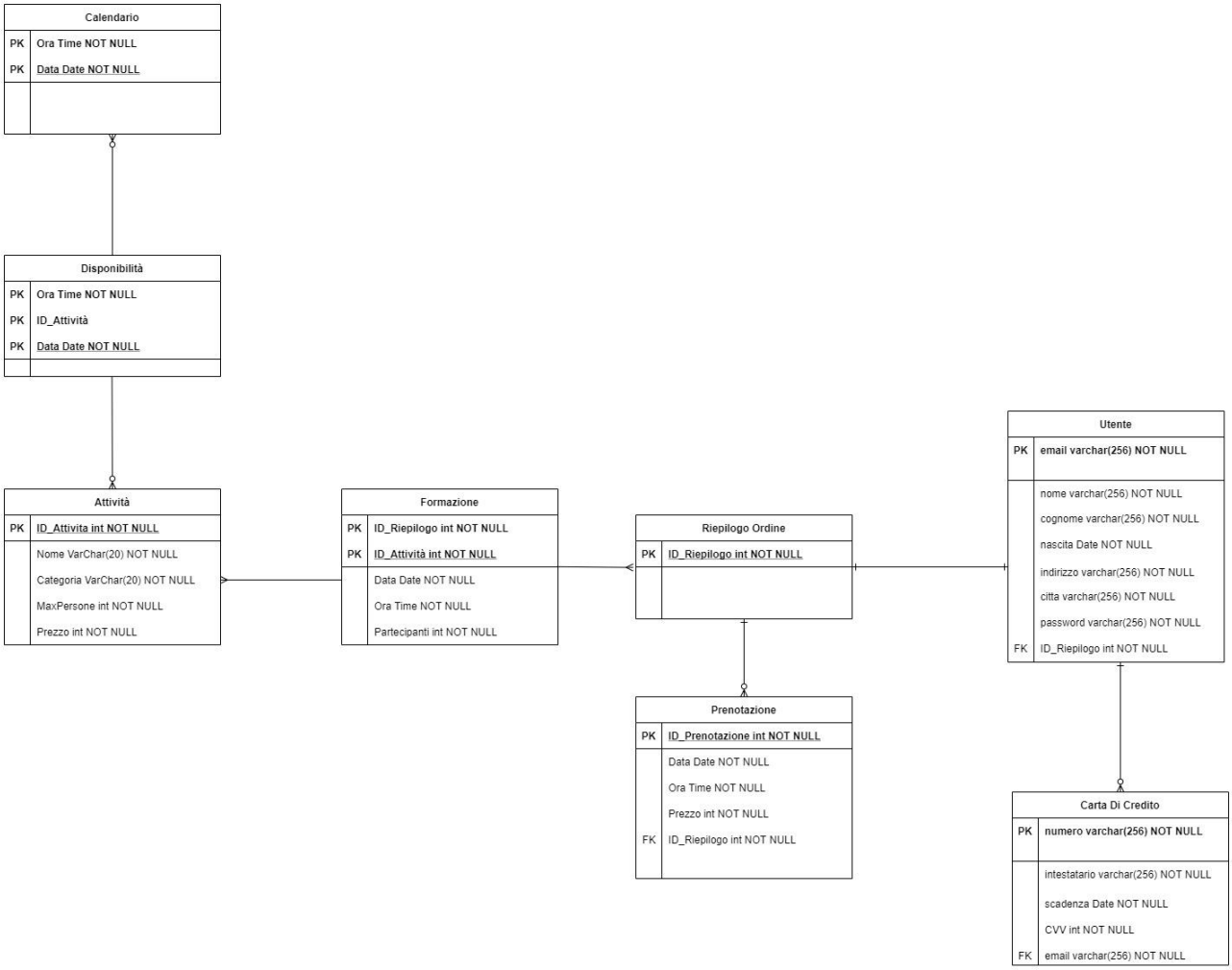
A livello di hardware, la webapp necessita di una macchina sulla quale poter installare i programmi che verranno usati per rendere accessibile il sito.

A livello di software, l'applicazione necessita del programma *Apache Tomcat* il quale consente di accedere al sito e di *MySQL* per realizzare e gestire il database.

Inoltre, la webapp e il database verranno installati sulla stessa macchina così da abbattere i rischi di fallimento durante il lancio dell'applicazione e i ritardi di comunicazione tra webapp e database.

3.4 Persistent Data management

Per tenere traccia dei dati che vengono utilizzati al fine del corretto funzionamento del sistema si utilizza un



database relazionale scritto con linguaggio SQL.

L'entità **Giorni Della Settimana** modella i giorni della settimana e gli orari di quando è disponibile l'attività. È definita dai seguenti attributi:

- Data
- Orario

È in una relazione molti a molti con l'entità **Attività.** L'entità attività rappresenta tutte le attività presenti che possono essere svolte ed è definita dai seguenti attributi:

- Nome attività
- Categoria Attività
- Max numero persone
- Prezzo

È in una relazione molti a molti con l'entità **Riepilogo Ordine** che rappresenta le attività che vengono selezionate dal cliente per essere successivamente prenotate. L'entità riepilogo ordine è definita dai seguenti attributi:

- ID_Riepilogo

L’entità riepilogo ordine è convolta in due relazioni: una prima relazione, uno a molti, con l’entità **Prenotazione,** e una seconda relazione, uno a uno, con l’entità **Utente.**

L’entità **Prenotazione** modella la prenotazione effettuata dal cliente ed è definita dai seguenti attributi:

- ID_Prenotazione
- Data Prenotazione
- Ora Prenotazione
- Metodo Pagamento

L’entità **Utente** rappresenta l’utente che è registrato al sito ed è caratterizzato dai seguenti attributi:

- Nome
- Cognome
- Data di nascita
- Citta di residenza
- Via
- Email
- Password

L’entità utente si trova in una relazione zero a molti con l’entità carta di credito.

L’entità **Carta Di Credito** rappresenta la carta di credito che il cliente associa al proprio account per effettuare il pagamento e confermare la prenotazione. È definita dai seguenti attributi:

- Numero carta
- Intestatario
- Scadenza
- Cvv

3.5 Controllo degli accessi e sicurezza

Il controllo degli accessi avviene tramite l'utilizzo di username e password per gli utenti, la password deve rispettare determini requisiti in modo da prevenire accessi non autorizzati e quindi salvaguardare i dati sensibili dell'utente. Per avere un sito sicuro che prevenga eventuali attacchi bisogna controllare e dare l'accesso soltanto ad utenti autorizzati questo avviene controllando che la pagina viene caricata solo se si è eseguito l'accesso, un altro modo per difendersi è quello della validazione dell'input per evitare *SQL injection* ovvero di inviare comandi al database senza controllo quando c'è un' eccezione non bisogna mai mostrare all' utente la causa dell'errore ma una pagina di default *error* , questo perché potrebbe essere utilizzato in seguito come vulnerabilità del sito.

Oggetto ***Attore***	**Attività**	**Prenotazione**	**Le Mie Carte**	**Riepilogo Ordine**
Utente non registrato	Visualizzazione			
Utente registrato	Visualizzazione	Creazione visualizzazione Rimozione	Aggiunta Rimozione	Visualizzazione Modifca
Gestore	Modifica Aggiunta Rimozione	Visualizzazione		

3.6 Controllo flusso globale del sistema

Il sistema che andremo sviluppare è web based, dunque sarà capace di gestire molte richieste contemporanee da diversi utenti.

Un gestore degli eventi è incaricato di raccogliere le interazioni, dunque le richieste, dell'utente e distribuirle ai controllori appropriati. In questo modo tutte le richieste generate vengono intercettate e gestite nel migliore dei modi.

3.7 Condizioni limite

3.7.1 Avvio del sistema

Identificativo	**Avviamento**	
Descrizione	Il sistema viene avviato dal gestore.	
Attore principale	Gestore	
Attore secondario	N.D.	
Condizione d'entrata	Il sistema non è ancora stato avviato.	
Condizione d'uscita con successo	Il sistema è online.	
Condizione d'uscita con fallimento	Il sistema è offline.	
Rilevanza	Molto alta	
Frequenza stimata	Una volta all'anno	
	Flusso di eventi principale	
	Gestore	Sistema
	1. Avvia il server. 2. Lancia il servizio del DBMS tramite l'apposito comando. 3. Esegue il web container.	
		4. Comunica al gestore che l'avviamento è andato a buon fine.
	Flusso di eventi d'errore	
		5. Comunica al gestore l'errore che si è verificato.

3.7.2 Terminazione

Identificativo	**Terminazione**	
Descrizione	Il sistema viene arrestato dal gestore.	
Attore principale	Gestore	
Attore secondario	N.D.	
Condizione d'entrata	Il gestore ha accesso alla macchina su cui il sistema è in esecuzione.	
Condizione d'uscita con successo	Il sistema è offline.	
Condizione d'uscita con fallimento	Il sistema è ancora online.	
Rilevanza	Molto alta	
Frequenza stimata	Una volta all'anno	
	Flusso di eventi principale	
	Gestore	Sistema
	1. Termina il servizio del web container chiudendo la shell in cui questo è avviato	
		2. Comunica al gestore che l'arresto è andato a buon fine.
	3. Termina il servizio del DBMS.	
		4. Comunica al gestore che il servizio è stato arrestato correttamente.
	Flusso di eventi d'errore	
		5. Comunica al gestore l'errore che si è verificato.

3.7.3 Fallimento

Il sistema può incorrere in diverse tipologie di fallimento collegate a problematiche di hardware o di software. In particolare:

- *Fallimento hardware*
  - *Crash del disco su cui i dati persistenti sono salvati*, il sistema non prevede alcuna strategia di backup per il recupero dei dati.
  - *Crash del server*, il sistema non prevede alcuna strategia per la continua operabilità del server.
- *Fallimento software*
  - *Impossibilità di stabilire una connessione con il database*, il sistema non prevede alcuna strategia di risoluzione ma si limita a segnalare l'errore.

3.8 Servizi dei sottosistemi

Al momento non abbiamo individuato servizi specifici dei sottosistemi.