BARBER APP

System Design Document (SDD)

Autori
Francesco Pio Guerriero matr. 0512114790
Luca Vincenzo Lambiase Fontana matr. 0512113338
Ivan Prota matr. 0512106063

Sommario

Introduzione

Obbiettivi del Sistema

L'applicazione Android per prenotazioni in un salone di parrucchieri consente agli utenti di prenotare appuntamenti per servizi di taglio e styling. Il sistema gestisce la registrazione degli utenti, la visualizzazione dei servizi disponibili, la prenotazione e la conferma degli appuntamenti.

Design Goals

Illustriamo nella seguente tabella gli obbiettivi di design per il sistema e le relative priorità (a numeri più bassi corrispondono priorità più elevate). Per ogni obbiettivo riportiamo anche l'origine, facendo riferimento, in particolare, all'identificativo del requisito non funzionale ad esso associato.

Priorità	ID	Descrizione	Categoria	Origine
1	DG_1	Progettare un'interfaccia utente intuitiva e accessibile per clienti ed impiegati di ogni età, garantendo disponibilità 24 ore al giorno.	Usabilità	RNF1
2	DG_2	Massimizzare l'affidabilità del sistema e minimizzare gli impatti degli errori. Se un'utente inserisce un input non valido il sistema non si deve bloccare, ma non deve accettare l'input e notificare l'utente invitandolo a correggere l'errore.	Affidabilità	RNF2
3	DG_3	Facilitare gli aggiornamenti e le modifiche future attraverso una progettazione ben organizzata.	Manutenibilità	RNF3
4	DG_4	Garantire un alto livello di sicurezza per proteggere le informazioni sensibili degli utenti. Questo implica proteggersi contro attacchi comuni come SQL Injection, utilizzare algoritmi di hashing robusti per memorizzare le password, implementare un sistema di autorizzazioni basate su ruoli.	Sicurezza	RNF4
5	DG_5	Garantire un elevato livello di portabilità su diverse piattaforme. Mantenere una struttura modulare per agevolare l'adattamento a differenti sistemi operativi.	Portabilità	RNF5

		Questo comporta sviluppare inizialmente per Android, con successiva revisione e adattamento per IOS.		

Trade-offs

Complessità vs. Usabilità

Il team si impegna a non implementare funzionalità troppo avanzate, le quali potrebbero aumentare la complessità dell'applicazione, influenzando negativamente l'usabilità. Il team si impegna a bilanciare la complessità delle funzionalità mantenendo un'esperienza utente semplice ed intuitiva.

Performance vs. Costo

Implementare ottimizzazioni per migliorare le prestazioni potrebbe aumentare i costi di sviluppo e manutenzione. Il team si impegna a valutare molto attentamente queste scelte, valutando se le prestazioni aggiuntive sono essenziali rispetto ai costi associati.

Sicurezza vs. Usabilità

Introdurre misure di sicurezza più rigorose potrebbe complicare il processo di autenticazione per gli utenti. Il team si impegna a trovare un'equilibrio tra sicurezza e facilità d'uso per evitare che le misure di sicurezza possano ostacolare l'accesso agli utenti rendendolo troppo complesso.

Scalabilità vs. Complessità

Progettare per una maggiore scalabilità può richiedere un aumento della complessità dell'architettura. Il team si impegna a valutare bene se la complessità aggiuntiva è giustificata dalle esigenze attuali e future del sistema in termini di crescita del carico di lavoro.

Innovazione vs. Stabilità

Introdurre nuove tecnologie o metodologie può comportare un rischio di instabilità del sistema. Valutare attentamente il rischio e l'impatto della introduzione di innovazioni rispetto alla stabilità complessiva del sistema.

Flessibilità vs. Specificità

Progettare un sistema altamente flessibile può richiedere l'aggiunta di configurazioni complesse. Il team ha definito, infatti, la compatibilità del sistema solo per piattaforme IOS ed Android in modo da evitare configurazioni eccessivamente complesse e specifiche che possono confondere gli utenti.

Panoramica

Nel documento verranno affrontati l'analisi delle architetture di sistemi simili, la scomposizione in sottosistemi del sistema proposto con la definizione della strategia di deploy e le condizioni limite. Verranno quindi definiti i servizi esposti da ciascun sottosistema.

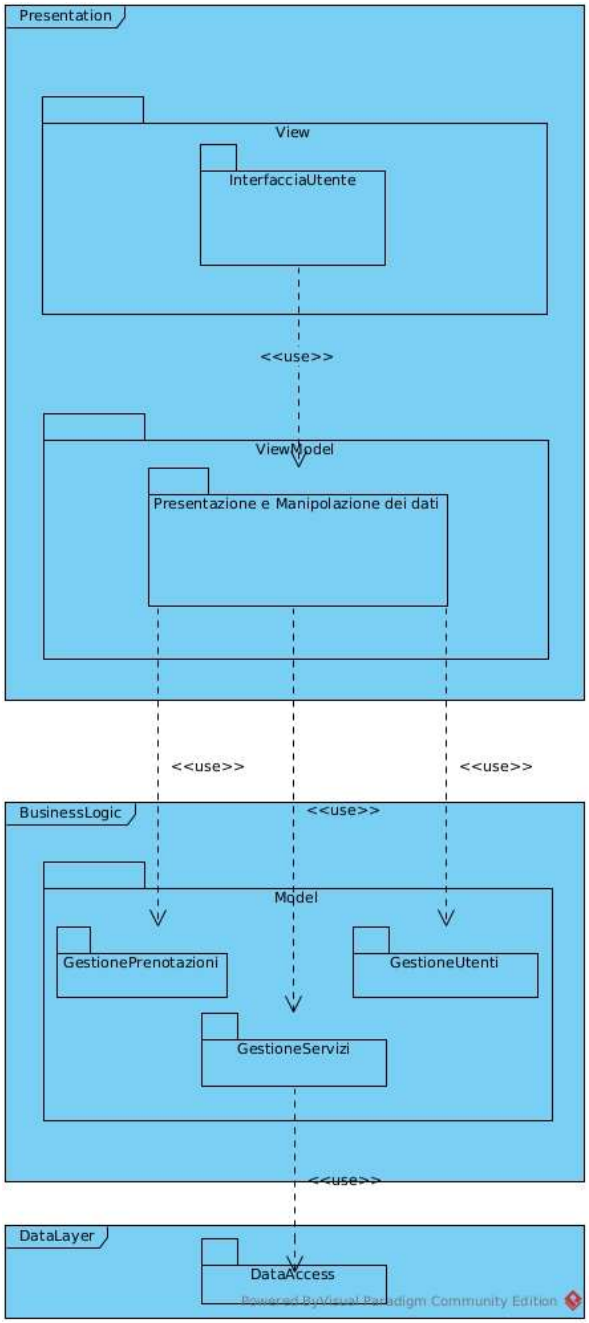
Architettura di sistema simili

Una piattaforma simile a BarberApp è SaloonAppy. SaloonAppy è probabilmente costituita da un'app android che funge da interfaccia utente per gli utenti finali, compresi clienti, parrucchieri e impiegati del salone. Essa ha un modulo di autenticazione, la quale gestisce la registrazione e l'autenticazione degli utenti. Molto probabilmente avrà una sezione, la quale visualizzerà i servizi e le prenotazioni, la quale presenta i servizi offerti dal salone e consente agli utenti di visualizzare gli orari disponibili e prenotare gli appuntamenti. Avrà sicuramente una sezione per la gestione del profilo, la quale consente agli utenti di aggiornare il proprio profilo, inclusi dettagli personali e preferenze. Il backend sarà strutturato nel modo in cui gestirà la logica di buisness, l'accesso al database e le operazioni di comunicazione tra frontend e database. Ad esempio, per quanto riguarda la gestione degli utenti, esso registrerà ed autenticherà gli utenti. Ed ovviamente implementerà anche ruoli come (cliente, parrucchiere, impiegato). Esso gestirà anche le prenotazioni, ovvero, verificherà la disponibilità degli orari, gestirà le richieste di prenotazione e invia conferme agli utenti. Poi ci sarà anche la gestione dei servizi, la quale manterrà una lista dei servizi offerti dal salone. Ed invece per quanto riguarda la sicurezza e autorizzazioni implementerà sicuramente misure di sicurezza come l'hashing delle password e l'autorizzazione basata su ruoli. Per quanto riguarda il database, invece, probabilmente esso sarà strutturato da tre tabelle, le quali manterranno i dati persistenti del sistema, le quali sono: Utenti, Servizi, Prenotazioni le quali archivieranno informazioni sugli utenti, i servizi offerti e gli appuntamenti prenotati. Ci saranno sicuramente delle API (interfacce di programmazione) le quali consentiranno la comunicazione tra frontend e backend, le quali probabilmente saranno divise in: API di autenticazione (gestiscono richieste di registrazione e autenticazione), API di Prenotazione (gestiscono le richieste di prenotazione e conferma), API di Servizi (Permettono al frontend di recuperare informazioni sui servizi disponibili). Mentre per quanto riguarda il sistema di hosting, il sistema sarà probabilmente ospitato su un server web, il quale può potenzialmente essere un servizio di cloud computing come AWS o Google Cloud, mentre la distribuzione dell'app sarà distribuita attraverso Google Play Store o AppStore.

Architettura del sistema proposto

Panoramica

L'architettura del sistema è suddivisa in tre componenti principali: Frontend (App Android), Backend (Server) e Database.



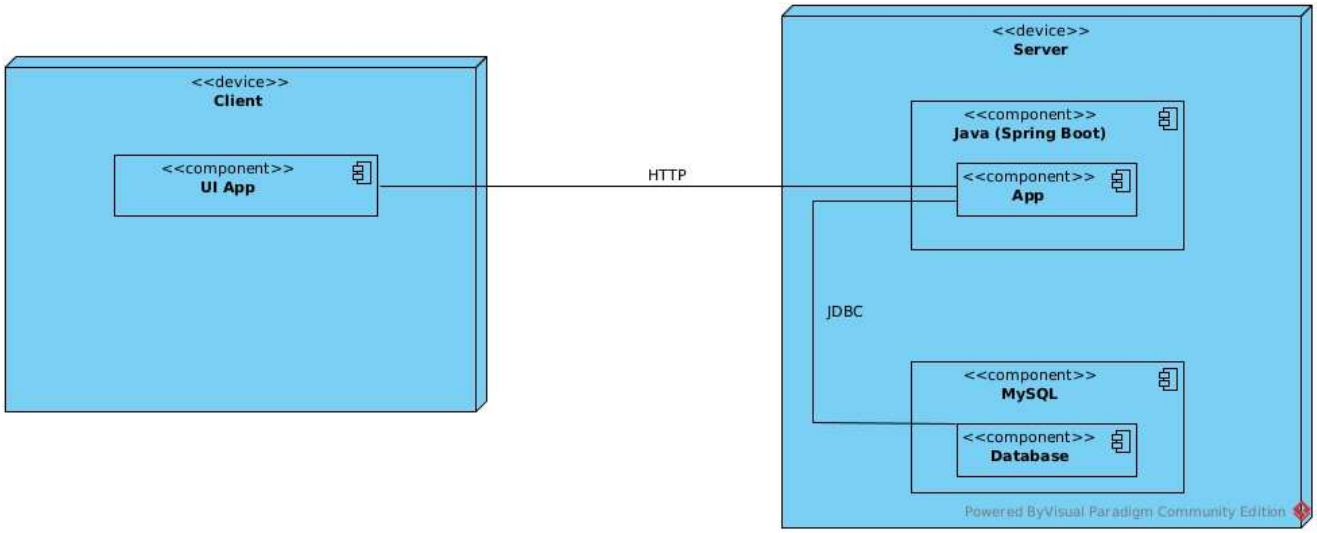
Si è deciso di basare BarberAPP su un'architettura MVVM (Model, View, ViewModel) in modo da avere un'architettura modulare, la quale struttura il codice in moduli ben definiti per migliorare la manutenibilità e la comprensibilità, in più ci permette di avere un sistema molto scalare considerando un potenziale aumento del numero di utenti e di prenotazioni. Vediamo il sistema suddiviso su tre livelli:

- **Il livello di presentation,** che contiene sia l'interfaccia utente che interagisce direttamente con l'utente. Risponde agli input utente e mostra i dati forniti dalla **ViewModel.** La ViewModel serve come intermediario tra la view ed il model. Prepara e fornisce i dati necessari alla view, gestisce gli eventi dell'utente e può contenere la logica di presentazione leggere, ma non la logica di buisness complessa.
- **Il livello di business logic,** questo livello contiene il model il quale, a sua volta, contiene tutta la logica di business dell'applicazione. Gestisce la manipolazione dei dati, la validazione e l'applicazione delle regole di business. Ed interagisce con il database per l'accesso ai dati.
- **Il livello data,** col sottosistema Data Access che ha l'accesso al database per salvare, prelevare o modificare dati.

Considerazioni aggiuntive: La ViewModel agisce da collegamento tra la presentation (View) e la logica di business (Model). Si occupa principalmente di preparare i dati in un formato adatto alla View e di gestire la comunicazione tra View e Model. La ViewModel può contenere logiche di presentazione come la formattazione dei dati visualizzati, la gestione di eventi specifici della UI, ma evita di contenere logiche di business complesse. Le decisioni di business vengono prese dal model.
La separazione tra Presentation e Business Logic consente di mantenere un'architettura più modulare e facilita la testabilità delle singole componenti. In breve, la ViewModel si colloca nella sezione di Presentation, ma collabora strettamente con la logica di business contenuta nel Model. Questa suddivisione di responsabilità contribuisce a mantenere l'applicazione ben organizzata e manutenibile.
Realizziamo quindi la View e la ViewModel al livello di presentazione, il model è localizzato totalmente al livello di Business Logic.

Hardware/software mapping

BarberApp consiste di un'applicazione distribuita installabile su qualsiasi server in grado di eseguire Java e MySQL. L'architettura scelta prevede l'interazione della piattaforma con un database: date le risorse si ritiene opportuno installare sia il server che il database in un unico nodo. Il sistema sarà accessibile tramite l'applicazione di UI installata su dispositivi android di ultima generazione a disposizione degli attori.



Gestione dei dati persistenti

Per la gestione dei dati persistenti si è deciso di utilizzare MySQL, questo per due motivi principali:

- I dati sono ben strutturati e per questo si prestano particolarmente ad essere memorizzati in questo DBMS
- Per abbattere ulteriormente i costi, strutturare dei file avrebbe richiesto maggiore lavoro e quindi più costi; inoltre, MySQL è gratuito e di conseguenza comporta un costo in meno.

Infine, ovviamente MySQL ci permette di trattare i dati ed effettuare operazioni con estrema semplicità e di aggiungere un ulteriore livello di sicurezza per l'accesso dei dati.
Le seguenti entità, provenienti dal Class Diagram verranno rese persistenti:

- Cliente
- Dipendente
- Titolare
- Appuntamento
- Servizio

Controllo degli accessi e sicurezza

Il controllo degli accessi è garantito tramite l'utilizzo di username e password per gli utenti del sistema che hanno possibilità di creare o modificare gli oggetti che modellano entità del dominio, cosi da prevenire accessi non autorizzati ad informazioni sensibili. Implementa un sistema di autenticazione sicuro, utilizzando un sistema di confronto tra hash. Quando un'utente si logga nel sistema attraverso le

sue credenziali, automaticamente ogni volta che un'utente tenta di fare un'operazione sul sistema, o richiede un servizio, all'interno della richiesta viene controllato che questo utente sia abilitato o meno a fare quella determinata operazione o richiedere quel determinato servizio. Questo sistema viene implementato, in modo tale da non interpellare ad ogni richiesta il database, per controllare se l'utente è autenticato o meno. Per quanto riguarda le password sul database ci impegniamo ad utilizzare algoritmi di hashing robusti come SHA512. Avremo un tipo di autorizzazioni basata su ruoli per garantire che gli utenti abbiano accesso solo alle risorse e alle azioni appropriate. Le operazioni che gli utenti dell'applicazione possono effettuare sugli oggetti sono riportate nella tabella che segue:

	Utenza	**Appuntamento**	**Servizi**	**Account Dipendenti**
Cliente	Registrazione Login Modifica profilo Cancellazione account	Prenotazione appuntamento Cancellazione appuntamento Visualizzazione appuntamento	Visualizzazione servizi	
Dipendente	Login Modifica profilo	Visualizzazione appuntamento	Visualizzazione servizi	
Titolare	Login Modifica profilo	Cancellazione appuntamento Visualizzazione appuntamento	Visualizzazione servizi Aggiunta servizi Cancellazione servizi Modifica servizi	Aggiunta dipendente Cancellazione dipendente

Controllo flusso globale del sistema

Il sistema adotta due controlli del flusso globale, uno di tipo event-driven, e uno di tipo thread-driven. Per quanto riguarda il tipo di controllo event-driven utilizziamo Spring Boot con Spring Integration, il quale è un framework che semplifica l'integrazione dei sistemi attraverso il paradigma event-driven. Mentre per quanto riguarda il controllo thread-driven, utilizziamo sempre Spring Boot con Spring Async, che permette la programmazione asincrona e concorrente. Spring Async ci consente di eseguire metodi in modo asincrono utilizzando un pool di thread gestito da Spring e questo ci permette di avere un'interazione concorrente tra l'applicazione e più client tramite l'intercettazione di eventi generati

proprio da quest'ultimi. Questa soluzione permette al sistema di poter rispondere a più utenti contemporaneamente.

Boundary Condition

Startup sistema

Nome caso d'uso	Startup sistema	
Attori partecipanti	Amministratore	
Flussi di eventi	**Amministratore**	**BarberApp**
	1. L'amministratore scrive in console "./startup.sh"	
		2. Il sistema inizia l'esecuzione creando titolare e dipendenti ed inizializzandoli
Pre-Condizioni	L'amministratore è connesso tramite una shell al server che ospita l'applicazione	
Post-Condizioni	L'applicazione è raggiungibile da qualsiasi dispositivo android che abbia installata l'applicazione di UI e che è connesso alla rete	

Shutdown Sistema

Nome caso d'uso	Shutdown Sistema	
Attori partecipanti	Amministratore	
Flussi di eventi	**Amministratore**	**BarberApp**
	1. L'amministratore scrive in console "./shutdown.sh"	
		2. Il sistema esegue lo script di shutdown
Pre-Condizioni	L'amministratore è connesso tramite una shell al server che ospita l'applicazione. L'applicazione è stata precedentemente avviata.	
Post-Condizioni	L'applicazione non è più raggiungibile, tutte le operazioni che proverà a fare l'utente dall'applicazione di UI, daranno l'errore di connessione al server	

Fallimento

BarberApp può incorrere in diversi casi di fallimento, riguardanti sia l'hardware che il software:

- Fallimenti Hardware: Crash del dico su cui i dati persistenti sono salvati: il sistema non prevede alcuna strategia di backup e ripristino dei dati
- Fallimenti nell'ambiente di esecuzione: Interruzione della fornitura elettrica al server: il sistema non prevede alcuna strategia che ne garantisca l'operabilità in questo tipo di condizione
- Fallimenti software: Impossibilità di stabilire una connessione con database: una schermata notifica l'errore all'utente

Subsystem services

Sottosistema	Descrizione
Utenza	Consente ad un Utente non registrato di effettuare la registrazione e a un Utente registrato di effettuare il login. Permette a quest'ultimo di accedere alla propria pagina utente per modificare i dati personali inseriti all'atto di registrazione ed inoltre è possibile eliminare definitivamente il proprio account.

Servizio	Descrizione
Registrazione	Consente ad un'utente non registrato di creare un account nel sistema
Login	Consente ad un cliente/dipendente/titolare di accedere al sistema tramite una coppia e-mail e password
Modifica profilo	Consente ad un cliente/dipendente/titolare di modificare le informazioni relative al proprio account
Cancellazione account	Consente ad un cliente registrato di eliminare il proprio account

Sottosistema	Descrizione
Appuntamento	Consente ad un cliente registrato di effettuare una prenotazione e quindi di inserire una nuova prenotazione all'interno del sistema. Talvolta può anche eliminarla se decide che non può adempire all'appuntamento, o visualizzarla. Mentre permette al titolare di cancellare e di visualizzare le prenotazioni.

Servizio	Descrizione
Prenotazione appuntamento	Consente ad un cliente registrato di inserire una prenotazione per un determinato servizio all'interno del sistema
Cancellazione appuntamento	Permette ad un cliente registrato/titolare di eliminare una prenotazione dal sistema
Visualizzazione appuntamento	Consente ad un cliente/titolare/dipendente di visualizzare una prenotazione nel sistema

Sottosistema	Descrizione
Servizi	Consente al titolare di aggiungere, modificare, visualizzare o cancellare servizi che sono i vari servizi offerti dal salone. Consente ad utente o a un dipendete di visualizzare invece i servizi offerti dal salone.

Servizio	Descrizione
Visualizzazione servizi	Consente ad un cliente registrato/dipendente/titolare di visualizzare i vari servizi offerti dal salone
Aggiunta servizi	Permette al titolare di aggiungere un nuovo servizio che offre il suo salone
Cancellazione servizi	Permette al titolare di eliminare un servizio che il suo salone non offre più
Modifica servizi	Permette al titolare di modificare il prezzo di un determinato servizio che offre il salone

Sottosistema	Descrizione
Account Dipendenti	Consente al titolare di aggiungere nuovi dipendenti al suo salone, quindi di registrare nuovi dipendenti nel sistema. Consente al titolare di eliminare dipendenti dal suo salone in precedenza registrati nel sistema.

Servizio	Descrizione
Aggiunta dipendente	Consente ad un titolare di registrare un dipendente nel sistema che dovrà lavorare nel salone
Cancellazione dipendente	Permette al titolare di eliminare un dipendente dal sistema in precedenza registrato nel sistema