Documentazione progetto

Brumana Matteo, Foster Timothy James, Rota Matteo

matricole n. 1053588,1053643, 1053065



Università degli Studi di Bergamo

Laurea Magistrale in Ingegneria Informatica

Corso di Informatica III

Modulo di Progettazione e Algoritmi (6 CFU)

Anno Accademico 2021/2022

**INDICE**

**ITERAZIONE 0**

**ANALISI DEL CONTESTO**

Per parrucchieri ed estetisti l'agenda cartacea rappresenta ancora lo strumento più diffuso per gestire gli appuntamenti; tuttavia, carta e penna sono talmente semplici e comodi che risulta difficile accorgersi degli svantaggi che portano, tra cui figurano volatilità, mancanza di standard e necessità di averli con sé per poterli utilizzare.

Questo progetto si pone l’obiettivo di creare un'agenda virtuale facile da utilizzare quanto quella cartacea ma in grado di eliminarne i problemi e di ampliarne i vantaggi, mirando a migliorare la gestione del lavoro di un qualsiasi salone, ad esempio tramite il suggerimento di slot temporali per i prossimi appuntamenti, e di aumentare la fidelizzazione dei clienti dello stesso, ad esempio tramite l’emissione di buoni sconto di compleanno o l’invio di reminder in vista degli appuntamenti fissati.

Il risultato degli sforzi progettuali sarà un prodotto versatile, facilmente configurabile, ancor più facilmente utilizzabile e soprattutto basato il più possibile su tecnologie cloud, che elimineranno il costo d’acquisto e manutenzione di hardware locale e renderanno il sistema sempre raggiungibile senza alcuno sforzo economico o di configurazione.

**REQUISITI**

Un app Android, pensata per gestori di negozi di acconciature, dovrà:

1. fornire una sezione in cui gestire gli appuntamenti con i/le clienti;
   1. tramite l’app dovrà essere possibile creare/modificare/eliminare appuntamenti;
      1. durante la creazione di un appuntamento, dovrà essere possibile specificare chi è il/la cliente interessato;
         1. durante la specifica del/della cliente interessato/a dall’appuntamento, dovrà essere possibile inserire dati su nuovi/e clienti a sistema [punto 2.1];
      2. durante la creazione di un appuntamento, dovrà essere possibile specificare i servizi richiesti dal cliente e far decidere al software la durata dell’appuntamento;
         1. il sistema dovrà conoscere le specifiche dei vari servizi erogabili;
   2. tramite l’app dovrà essere possibile visionare gli appuntamenti creati su un calendario;
   3. un servizio esterno all’app dovrà inviare automaticamente via SMS ai/alle clienti un reminder relativo al loro appuntamento 24h prima dello stesso;
2. fornire una sezione in cui gestire le anagrafiche dei/delle clienti;
   1. tramite l’app dovrà essere possibile inserire/modificare/eliminare dati sui/sulle clienti;
   2. tramite l’interfaccia dovrà essere possibile consultare i dati registrati sui/sulle clienti;
3. fornire una sezione in cui gestire i buoni sconto emessi in occasione dei compleanni dei/delle clienti;
   1. un servizio esterno all’app dovrà occuparsi di generare automaticamente buoni sconto in occasione dei compleanni dei/delle clienti;
   2. un servizio esterno all’app dovrà occuparsi di eliminare automaticamente buoni sconto quando la loro data di scadenza risulta superata;

**STUDIO DI Fattibilità**

Dopo numerose discussioni, si è deciso di concretizzare il progetto in un’app Android che scambiasse informazioni con un database nel cloud, coadiuvata da funzioni periodiche ospitate sul cloud che gestissero i rapporti con la clientela.

Per realizzare tutto questo, si è dovuti andare alla ricerca di:

* Database cloud.

La scelta è ricaduta su MongoDB Atlas, database distribuito ospitato totalmente in cloud.

MongoDB è un database NoSQL basato su documenti, ciò comporta dei requisiti maggiori per il livello di applicazione, ma consente di distribuire le serie di dati e i processi di lavoro tra più server, rendendolo **scalabili in maniera** quasi **illimitata, garantendo anche un certo livello di** disponibilità e flessibilità dei dati.

In termini di progetto, il DB deve essere in grado di ospitare e fornire in tempo reale tutti i dati relativi a clienti, appuntamenti e buoni sconto per permettere la corretta esecuzione di tutte le funzionalità previste in fase di progettazione;

* Ambiente di esecuzione in cloud.

Amazon Web Services mette a disposizione alcuni servizi che permettono di usufruire di potenza computazionale tramite cloud computing, tra cui le Lambda Functions, che permettono di eseguire codice scritto in diversi linguaggi di programmazione per qualsiasi tipo di applicazione senza effettuare il provisioning o gestire server.

Per il progetto sono state implementate due Lambda Function, una per la generazione dei buoni sconto e una per i reminder al cliente.

* Connettori tra app Android e database cloud.

Il collegamento dell’applicazione Android al database MongoDB Atlas viene effettuato utilizzando l’API Realm sviluppata da MongoDB per Java, che offre l’accesso a tutte le operazioni, di base e non, con il database.

* Connettori tra server cloud e database cloud;

Il driver MongoDB nativo per Node.JS è una dipendenza che consente all’applicazione Javascript di interagire nel cloud tramite MongoDB Atlas.

Questi driver sono oggetti che forniscono un punto di contatto con le api di MongoDB Atlas e permettono di eseguire numerosi tipi di operazioni sui dati memorizzati in esso.

Sono comodi perché vengono operati come normali oggetti di Javascript e possono essere integrati facilmente nel codice.

* Servizi cloud di notifica SMS.  
  Twilio è una piattaforma che permette l’invio di messaggi SMS, utilizzando una REST API servita su HTTPS.

**TOOLCHAIN**

Per la realizzazione del software verranno utilizzati i seguenti strumenti:

|  |  |
| --- | --- |
| **Obiettivo** | **Strumenti** |
| Design | Astah UML |
|  | Carta e penna |
|  | Word |
| Sviluppo codice | Java (Android Studio) |
|  | Node.js (AWS) |
| Analisi del codice | JUnit (dinamica) |
|  | STAN4J (statica) |
|  | Testing dinamici manuali |
| Distribuzione e versionamento | GitHub |
| Documentazione | Word |

**USE CASES**

**UC1: Gestione appuntamenti**

*Descrizione*: il parrucchiere deve poter essere in grado di gestire gli appuntamenti fissati con i clienti: questo comprende avere la possibilità di inserire, modificare o eliminare appuntamenti e di poterli visionare su un calendario. Il sistema deve inviare un reminder ai clienti aventi un appuntamento 24h prima dello stesso.

*Attori coinvolti*: parrucchiere, trigger temporale.

*SubCases*:

* UC1.1: Creazione appuntamenti;
* UC1.2: Visualizzazione appuntamenti;
* UC1.3: Eliminazione appuntamenti;
* UC1.4: Modifica appuntamenti;
* UC1.5: Invio automatico reminder appuntamenti;

**UC2: Gestione clienti**

*Descrizione*: il parrucchiere deve avere a disposizione un archivio contenente dati sui propri clienti: questo comprende avere la possibilità di inserire, modificare o eliminare dati anagrafici e di poterli visionare.

*Attori coinvolti*: parrucchiere.

*SubCases*:

* UC2.1: Creazione anagrafiche clienti;
* UC2.2: Visualizzazione anagrafiche clienti;
* UC2.3: Eliminazione anagrafiche clienti;
* UC2.4: Modifica anagrafiche clienti;

**UC3: Gestione buoni sconto compleanno**

*Descrizione*: il sistema deve generare e inviare buoni sconto in occasione dei compleanni dei clienti. Il parrucchiere deve avere a disposizione un resoconto dei buoni in circolazione.

*Attori coinvolti*: parrucchiere, trigger temporale.

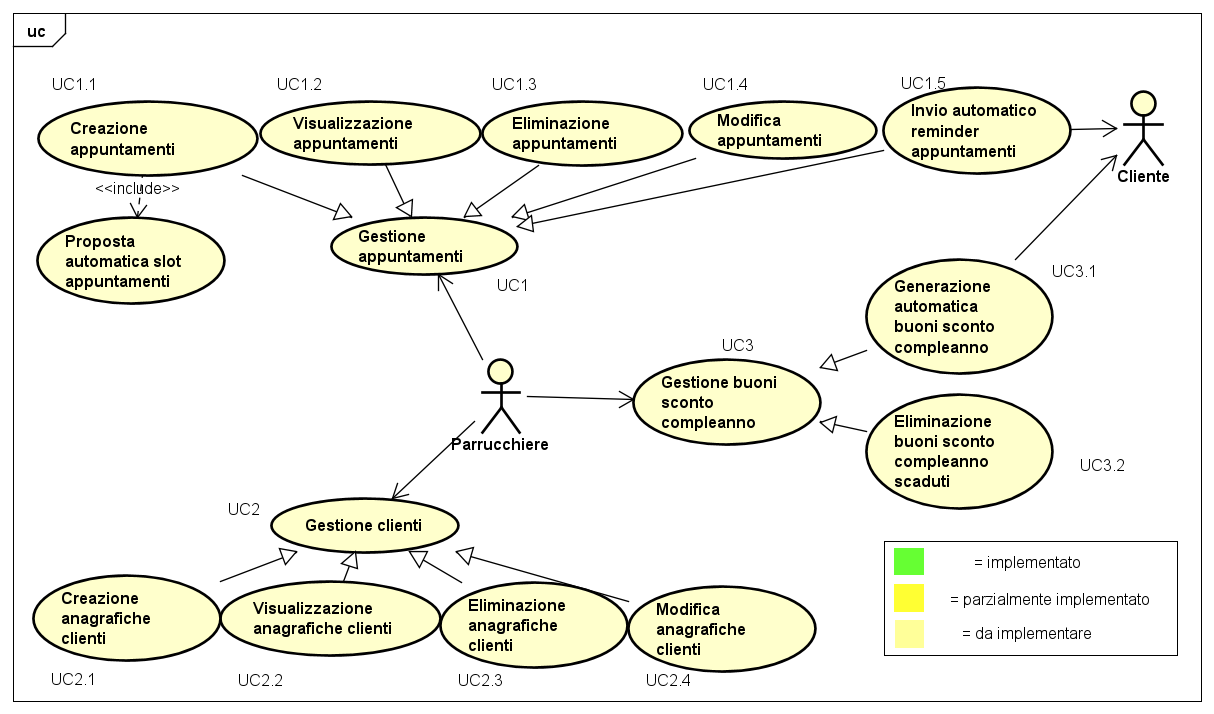
*SubCases*:

* UC3.1: Generazione automatica buoni sconto compleanno;
* UC3.2: Eliminazione automatica buoni sconto compleanno scaduti;

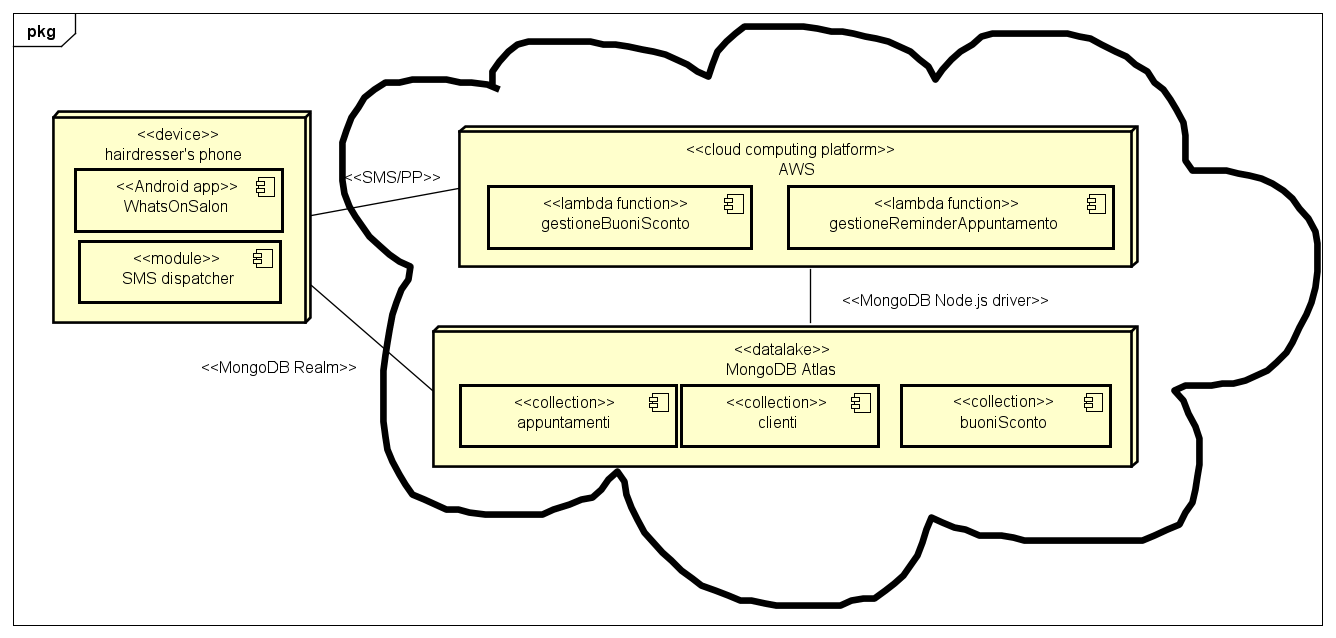
**CODA DELLE PRIORITÀ**

|  |  |
| --- | --- |
| **Codice** | **Titolo** |
| UC1 | Gestione appuntamenti |
| UC3 | Gestione buoni sconto compleanno |
| UC2 | Gestione clienti |

**USE CASE DIAGRAM**



**DEPLOYMENT DIAGRAM**



Per bibliografia:

<https://docs.mongodb.com/realm/sdk/>

<https://docs.mongodb.com/realm/sdk/android/>

<https://docs.mongodb.com/drivers/node/current/>

<https://www.twilio.com/docs/sms/send-messages>

<https://www.twilio.com/docs/sms/api>