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.

**STUDIO DI Fattibilità**

**SONO ARRIVATO QUA - MATTE**

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 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, una piattaforma di dati distribuita progettata per il 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 quasi **scalabili in maniera 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 ed appuntamenti per permettere la corretta esecuzione di tutte le funzionalità previste in fase di progettazione;

* Server che offrissero potenza computazionale in cloud.

Migrare da locale al cloud è il modo migliore per aumentare le potenzialità di un’applicazione ma non tutti hanno le possibilità di acquistare, possedere e mantenere i data center e i server fisici.

Amazon Web Services mette a disposizione alcuni servizi tecnologici che permettono di disporre di potenza computazionale tramite cloud computing.

Durante il progetto è stato implementato un database completamente gestito e flessibile in AWS con MongoDB Atlas.

Inoltre tramite il servizio AWS Lambda basato su eventi serverless, è possibile eseguire codici 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, sviluppata in Android Studio, al database MongoDB viene effettuato utilizzando un’API Restful di NodeJS;

<https://stackoverflow.com/questions/66982138/connecting-android-application-using-nodejs-restful-api-to-local-mongodb-databas>

**Mongoose può essere un esempio???????**

* Connettori tra server cloud e database cloud;
* Connettori tra server cloud e cliente;

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 remainder 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. Tramite l’app dovrà essere possibile visionare i buoni sconto in circolazione;

TOOLCHAIN

Per la realizzazione del software verranno utilizzati i seguenti strumenti:

|  |  |
| --- | --- |
| **Obiettivo** | **Strumenti** |
| Design | Astah UML |
|  | Carta e penna |
|  | Word |
| Sviluppo codice | Java (+ Swing) |
|  | Eclipse |
| Analisi del codice | JUnit (dinamica) |
|  | STAN4J (statica) |
| Distribuzione e versionamento | GitHub |
| Documentazione | Word |

CASI D'USO

**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 remainder 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 remainder 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: Visualizzazione buoni in circolazione;

**CODA DELLE PRIORITÀ**

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