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**

1. **Iterazione 0****3**

* 1. *Analisi del contesto 4*
  2. *Requisiti 4*
  3. *Studio di fattibilità 5*
  4. *Toolchain 6*
  5. *Casi d'uso 6*
  6. *Architettura hardware 8*

1. **Iterazione 1****9**

* 1. *Architettura software 10*

1. **Iterazione 2****12**

* 1. *xxxxxxxxxxxxxx 13*

**ITERAZIONE 0**

**1.1 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.

**1.2 REQUISITI**Un sistema pensato per titolari di attività per la cura della persona dovrà:

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

**1.3 STUDIO DI Fattibilità**

Dopo numerose discussioni, si è deciso di implementare il sistema descritto nei requisiti mediante un’app Android, supportandola con un database cloud e funzioni cloud.

Per realizzare tutto questo, sono state selezionate le seguenti tecnologie:

* Database cloud:  
  La scelta, data la familiarità con esso acquisita durante il corso di Tecnologie Cloud e Mobile, è ricaduta su MongoDB Atlas, database distribuito totalmente in cloud realizzato con tecnologia MongoDB (NoSQL, document-based).   
  La particolare struttura di questo database comporta dei requisiti maggiori per il livello di applicazione, ma consente di distribuire dati e processi su più server, rendendolo **scalabili in maniera** quasi **illimitata e garantendo** disponibilità e flessibilità dei dati;
* Connettore tra app Android e database cloud:  
  Il collegamento tra applicazione e database viene comodamente offerto dall’API Realm di MongoDB per ambiente Java, che offre l’accesso a tutte le operazioni effettuabili sul database mediante oggetti facilmente manipolabili. La scelta in questo ambito è dunque ricaduta su questa tecnologia.
* Ambiente di esecuzione in cloud:   
  Per questo servizio è stato scelto Amazon Web Services, visto anch’esso durante il corso di Tecnologie Cloud e Mobile, e in particolare la sua interfaccia Lambda Functions, che permette di sfruttarne la potenza computazionale eseguendo codice scritto in diversi linguaggi di programmazione in ambiente preconfigurato.
* Connettori tra ambiente di esecuzione cloud e database cloud:  
  Il driver MongoDB nativo per Node.js, su cui è ricaduta la scelta, è una dipendenza che consente alle applicazioni che lo sfruttano di interagire con MongoDB Atlas mediante oggetti che forniscono un punto di contatto con le API di MongoDB e permettono di eseguire numerosi tipi di operazione sui dati presenti nel database.
* Servizio di notifica SMS:  
  È stata scelta Twilio, REST API servita su HTTPS che permette l’invio di messaggi SMS;

**1.5 TOOLCHAIN**  
Viste le considerazioni riportate nello studio di fattibilità, è stata determinata la seguente toolchain per il progetto:

|  |  |
| --- | --- |
| *Obiettivo* | *Strumenti* |
| Progettazione | Astah UML |
|  | Carta e penna |
|  | Word |
| Sviluppo codice | Java (per app Android) |
|  | Node.js (per Lambda Functions) |
| Analisi del codice | JUnit (dinamica per app Android) |
|  | STAN4J (statica per app Android) |
|  | Test dinamici manuali (per Lambda Functions) |
| Distribuzione e versionamento | GitHub |
| Documentazione | Word |

**1.6 CASI D’USO**

UC1: Gestione appuntamenti

*Descrizione*: il titolare 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*: titolare, 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 titolare 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*: titolare.

*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.

*Attori coinvolti*: trigger temporale.

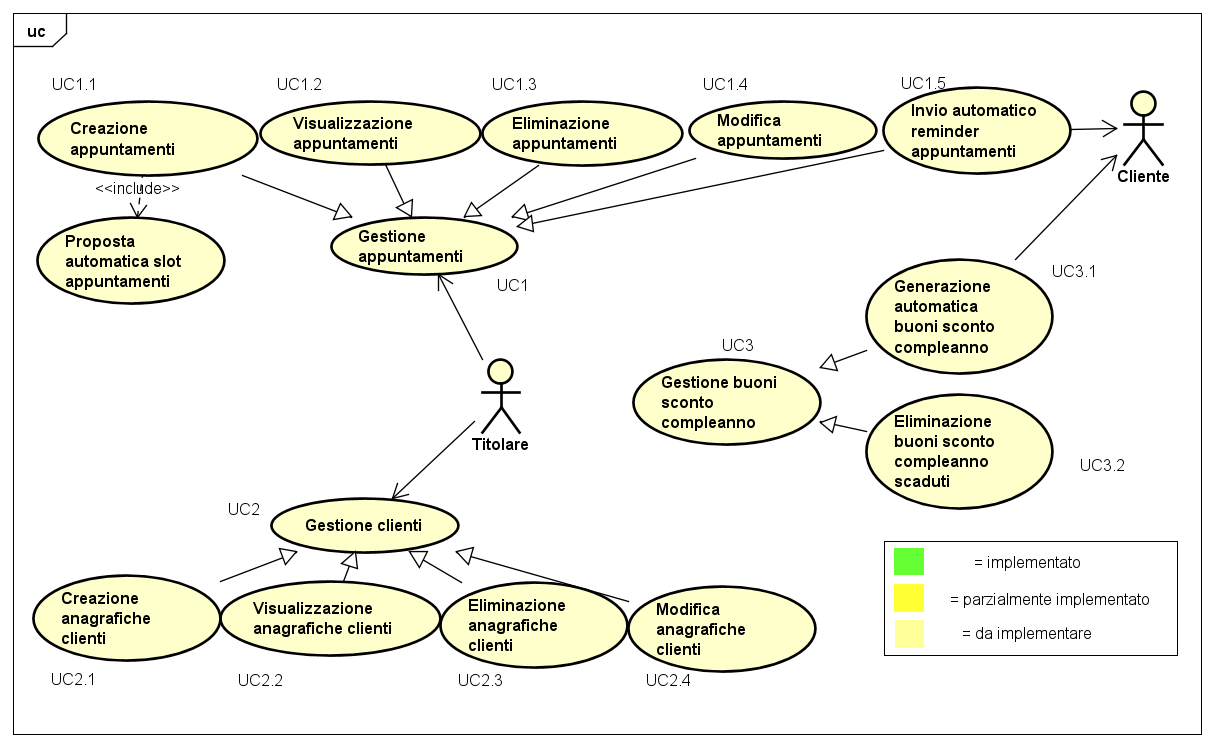
*SubCases*:

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

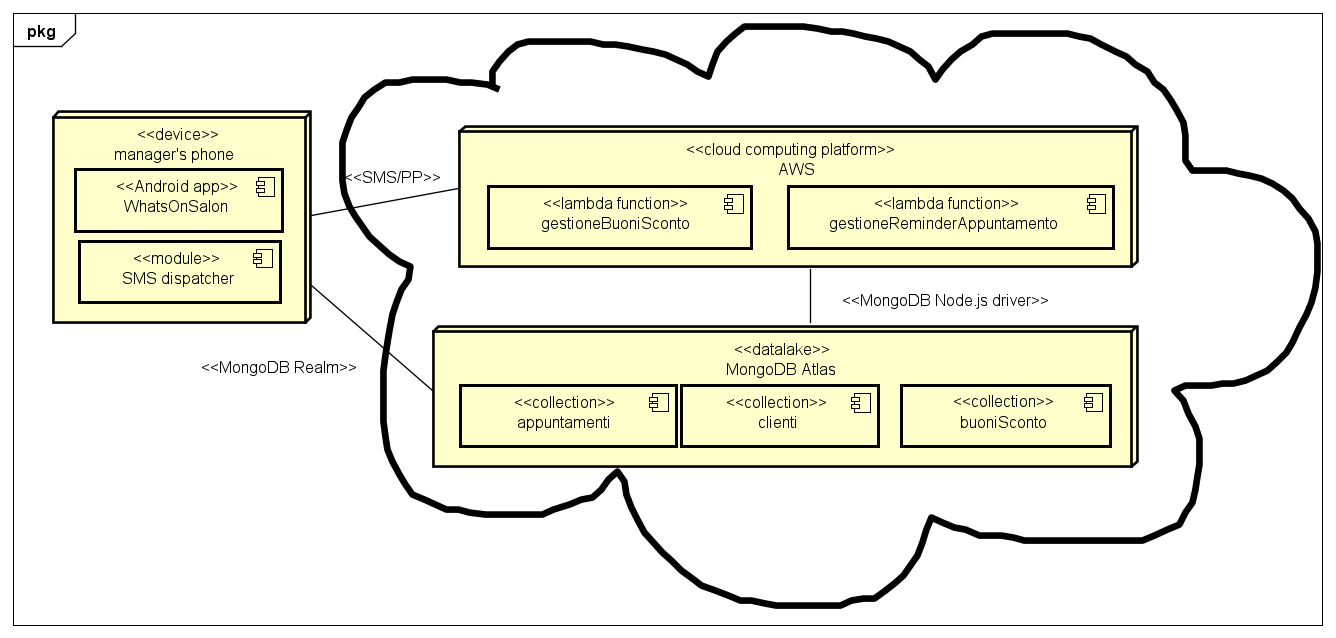
Una volta individuati i casi d’uso, questi sono stati ordinati per priorità di implementazione come segue:

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

I casi d’uso individuati sono inoltre stati mappati su un diagramma UML dei casi d’uso:



**1.6 ARCHITETTURA HARDWARE**Sulla base di quanto discusso in precedenza, è stata elaborata una prima bozza di possibile architettura del sistema ed è stata mappata su un diagramma UML, più precisamente un deployment diagram.  
La nuvola presente nel diagramma, ovviamente, rappresenta il cloud e le connessioni tra i vari dispositivi hardware esplicitano il tipo di tecnologia utilizzata per implementarle.

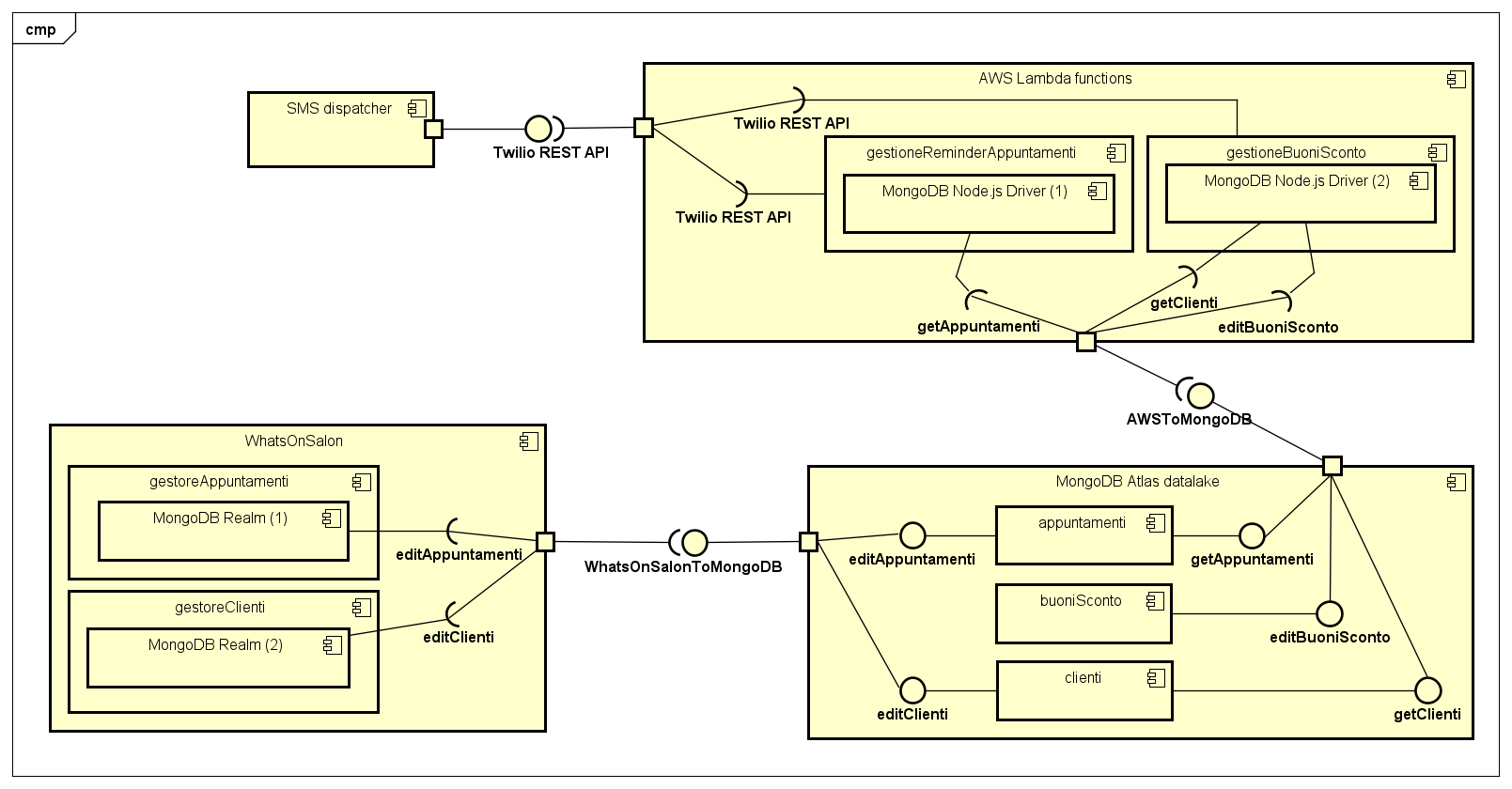


**ITERAZIONE 1**

**ARCHITETTURA SOFTWARE**

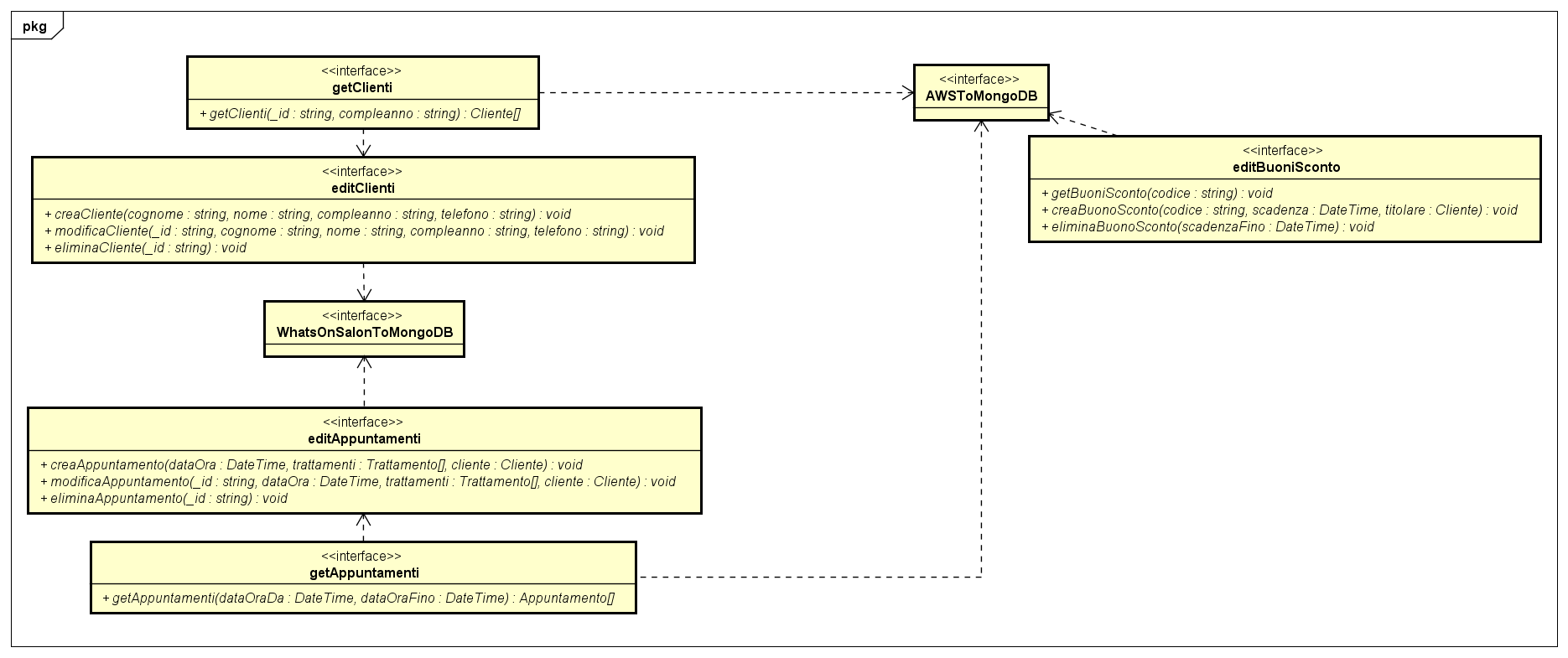
Basandosi sul lavoro svolto nell’iterazione precedente, è stato possibile cominciare a progettare l’architettura del software, riassunta in questo diagramma UML delle componenti.

Il diagramma ha lo scopo di rappresentare la struttura interna del sistema software modellato in termini delle sue componenti principali e delle relazioni fra esse. Per facilitarne la comprensione, in esso è stato fatto largo uso di porte e delegation connectors e dove era stato previsto l’utilizzo di software preesistente questo è stato aggiunto direttamente come subcomponent (es. MongoDB Realm) o interfaccia (es. Twilio REST API).

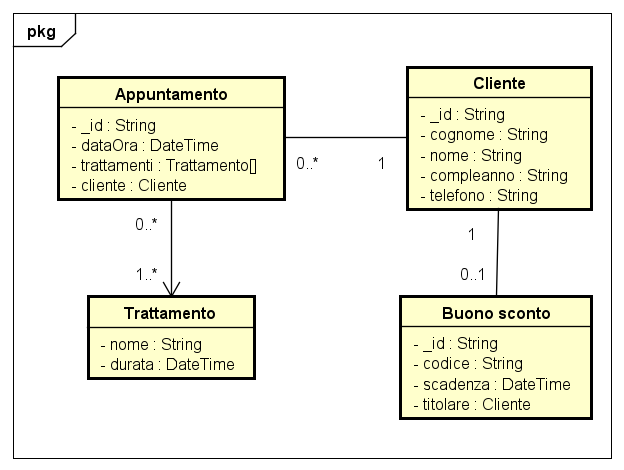


Le interfacce user-defined presenti nel component diagram sono esplicitate in quest’altro schema UML, che prende il nome di interface diagram.

Come si può osservare, esistono relazioni di ereditarietà tra le interfacce: ogni connessione tra component o subcomponent (visibile nel component model) è infatti realizzata da un’interfaccia che include dentro di sé tutte le interfacce richieste/offerte dalle parti in gioco.  
I metodi specificati all’interno di ogni interfaccia nello schema sono astrazioni di quella che diverrà la reale implementazione durante le fasi successive di sviluppo.



I tipi di dato user-defined presenti nell’interface diagram, che sono anche quelli che popoleranno il database, e le relazioni che intercorrono tra essi sono esplicitati in questo data diagram che, data la particolare architettura venuta a configurarsi in questo caso, è equivalente e funge anche da data transfer model.



**ITERAZIONE 2**

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>