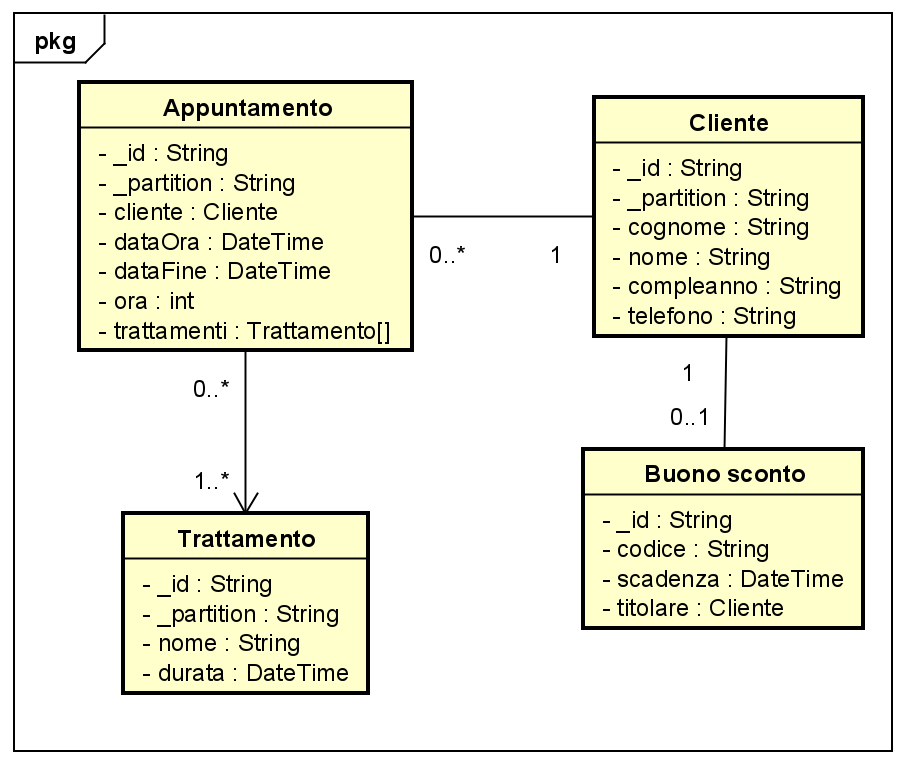
**ITERAZIONE 2**

**INTRODUZIONE**

Durante la preparazione per l’implementazione dei casi d’uso descritti nel paragrafo successivo si è resa necessaria una variazione all’interno del data model, causata dalla necessità di Realm di avere un campo \_partition in ogni documento nel database per poterlo recuperare dall’app, dalla comodità di avere un campo dataFine per ogni Appuntamento e dall’inaspettata necessità di dover avere l’orario di inizio di ogni Appuntamento sotto forma di intero per poter effettuare un ordinamento tra Appuntamenti.

Qui di seguito è riportato il data model che caratterizzerà l’elaborato da qui in avanti.

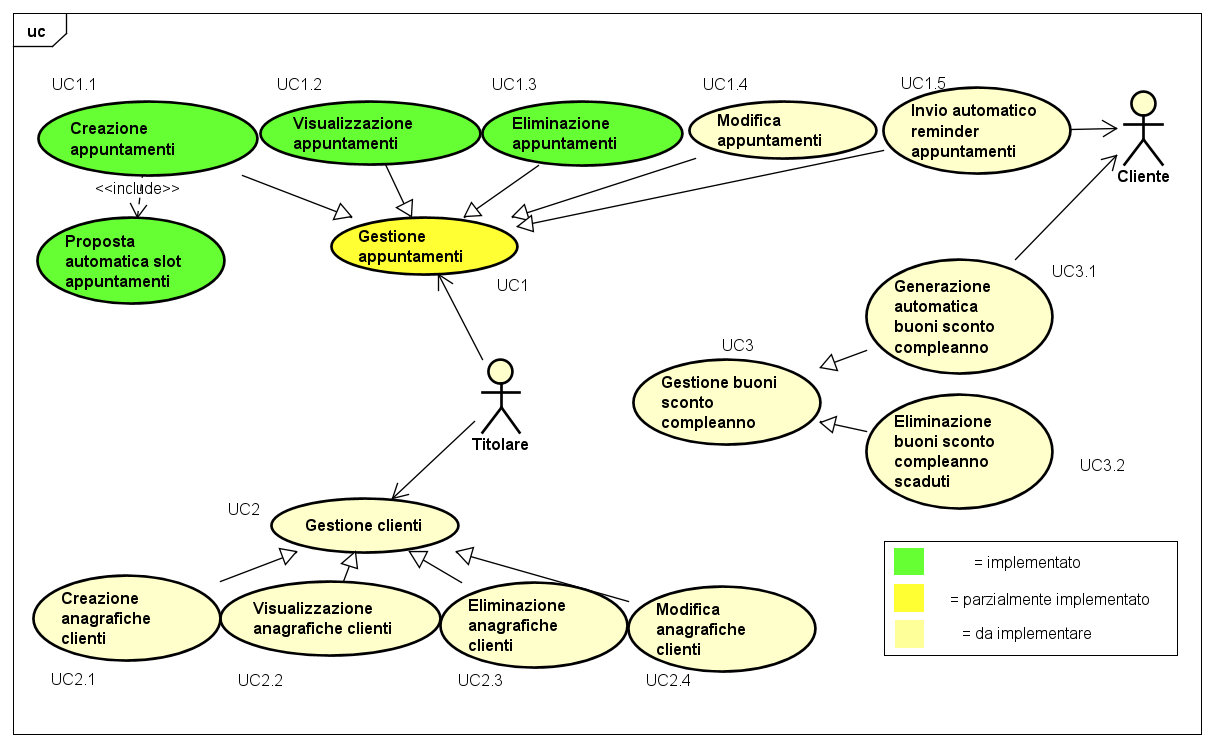


**CASI D’USO SELEZIONATI**

Nella seconda iterazione si è deciso di implementare i seguenti casi d’uso:

* UC1: Gestione appuntamenti (astratta):
* UC1.1: Creazione appuntamenti;
* UC1.2: Visualizzazione appuntamenti;
* UC1.3: Eliminazione appuntamenti;

Dal diagramma dei casi d’uso sottostante è possibile analizzare l’avanzamento del progetto facendo riferimento alla legenda in basso a destra.



UC1.1: Creazione appuntamenti *Descrizione*: il titolare deve poter essere in grado di fissare appuntamenti sul “calendario virtuale” fornito dall’app, dettagliando per ogni appuntamento cliente interessato (che deve essere inseribile a sistema sul momento, nel caso in cui non fosse già inserito), trattamenti previsti e data e ora nelle quali l’appuntamento è fissato, specificando queste ultime manualmente oppure scegliendole tra le proposte fornite dal sistema.

*Attori coinvolti*: titolare, algoritmo di proposta automatica slot appuntamenti.   
  
*Trigger:* richiesta di creazione appuntamento previa specifica dei dati ad esso relativi?.  
 *Postcondizione:* appuntamento avente i dati specificati presente sull’apposito database. *Procedimento:* 1. il titolare apre l’interfaccia di creazione appuntamenti;  
2. il titolare seleziona cliente interessato (eventualmente inserendone l’anagrafica a sistema, nel caso in cui questo non fosse già registrato), i trattamenti previsti e la data e l’ora in cui l’appuntamento è previsto, selezionandole manualmente o scegliendo una delle proposte provenienti dall’algoritmo di proposta;  
 3. il titolare richiede la creazione dell’appuntamento appena dettagliato;  
 4. il titolare viene riportato alla vista del calendario appuntamenti;

UC1.2: Visualizzazione appuntamenti *Descrizione*: il titolare deve poter essere in grado di visualizzare in un formato agile tutti gli appuntamenti relativi a un giorno da lui indicato.

*Attori coinvolti*: titolare.  
  
*Trigger:* selezione data.  
  
*Postcondizione:* stampa a video appuntamenti previsti per la data selezionata in un formato facilmente interpretabile.  
  
*Procedimento:* 1. il titolare seleziona una data in un’interfaccia stile calendario;  
 2. l’interfaccia viene riempita con gli appuntamenti previsti per la data selezionata, in un formato facilmente interpretabile;

UC1.3: Eliminazione appuntamenti *Descrizione*: il titolare deve poter essere in grado di eliminare appuntamenti fissati in precedenza.

*Attori coinvolti*: titolare.  
  
*Trigger:* richiesta di eliminazione di uno o più appuntamenti precedentemente selezionati.  
  
*Postcondizione:* appuntamenti di cui è stata richiesta l’eliminazione non più presenti nel database.  
  
*Procedimento:* 1. *Procedimento UC1.2* + checkbox di selezione per ogni appuntamento;  
 2. il titolare seleziona gli appuntamenti che intende eliminare;  
 3. il titolare richiede l’eliminazione degli appuntamenti selezionati;  
 4. il titolare viene riportato alla vista del calendario appuntamenti;

**PSEUDOCODICE**UC1.1: Creazione appuntamenti

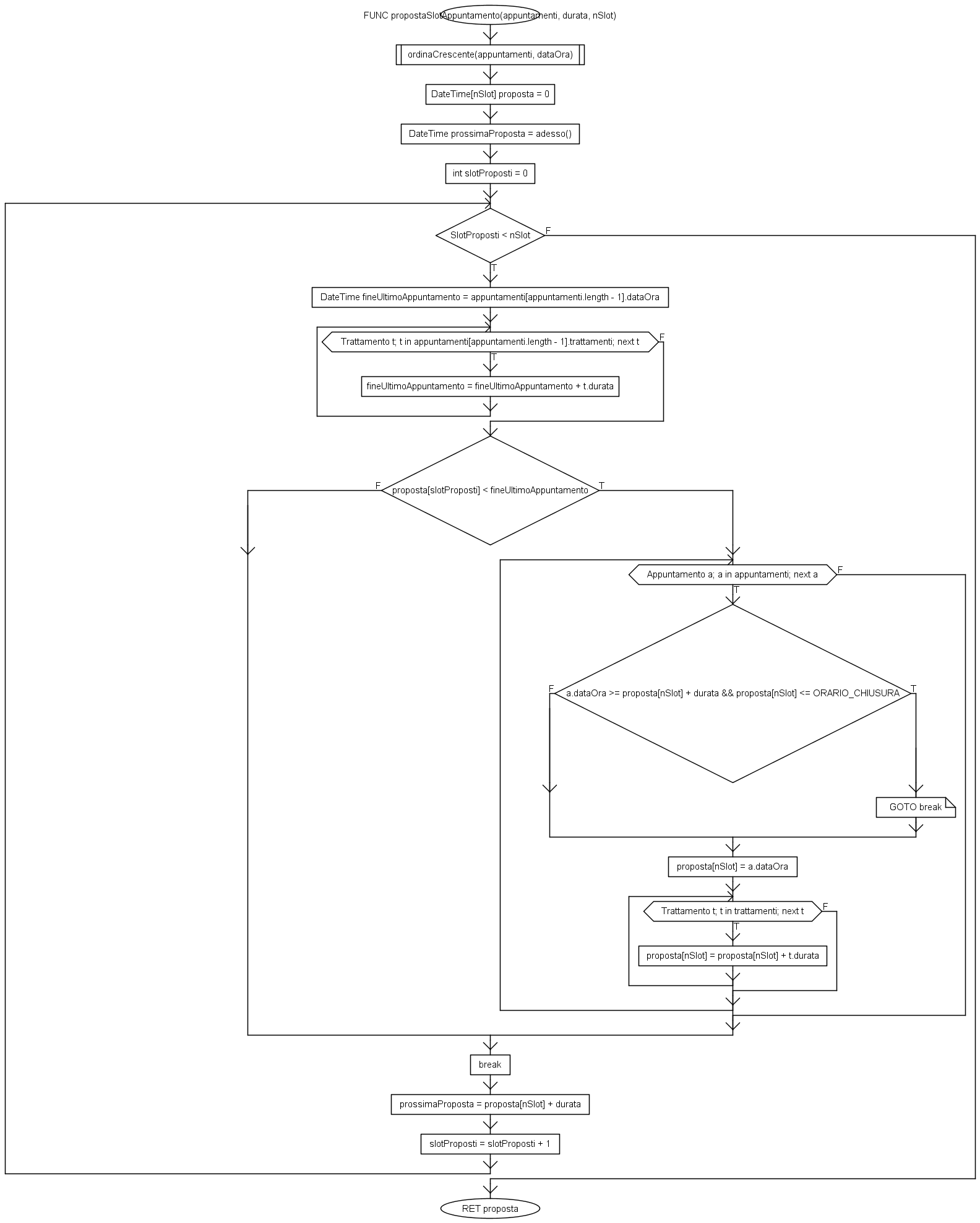
Immagine che contiene testo

Descrizione generata automaticamente

La complessità dell’algoritmo è Ω(nSlot), caso in cui tutte le proposte vengono trovate al primo tentativo e O(nSlot \* nAppuntamentiFuturi), caso in cui viene raggiunta una prossimaProposta successiva alla fine dell’ultimo appuntamento futuro solo nel corso dell’ultima iterazione del while.

Fortunatamente, tutti gli onerosi calcoli degli orari di fine appuntamento presenti nello pseudocodice sono stati semplificati dall’introduzione nel data model degli appuntamenti di un campo dataFine.

Si veda di seguito il flow chart dell’algoritmo.

****

**RISULTATI implementazione**



**ANALISI STATICA**

Per l’analisi statica del codice è stato utilizzato un plug-in disponibile in Android Studio chiamato MetricsReloaded che fornisce le metriche del codice sorgente in esame.  
Il plugin fornisce delle misure di complessità del codice. In particolare:

* **CogC:** Cognivite Complexity  
  Misura la complessità in termini di comprensione del codice. Questo valore aumenta all’aumentare del numero di strutture controllate e strutture annidate.
* **ev(G)**: Essential Cyclomatic Complexity   
  Misura quanto sia mal strutturato il flusso di controllo di un metodo.
* **iv(G)**: Design Complexity  
  Calcola la complessità di progettazione di un metodo. La complessità della progettazione è correlata al modo in cui un flusso di controllo dei metodi è interconnesso con le chiamate ad altri metodi.
* **v(G)**: Cyclomatic Complexity  
  È una misura del numero di percorsi di esecuzione distinti attraverso ciascun metodo. Questo può anche essere considerato come il numero minimo di prove necessarie per esercitare completamente il flusso di controllo di un metodo. In pratica, questo è 1 + il numero di if, while, for, do, switch case, catch, espressioni condizionali, and e or nel metodo.

L’analisi con MetricsReloaded ha dato i seguenti risultati, ordinati in base al valore decrescente di v(G).

Immagine che contiene testo

Descrizione generata automaticamente

**com.example.whatsonsalon.SelezionaSlotAppuntamento.proponiSlot(syncConfiguration)**

Per questo metodo si hanno alti valori per Cognitive Complexity, Design Complexity e Cyclomatic Complexity, giustificati dalla presenza di strutture di controllo annidate e chiamate ad altri metodi.

Per interpretare al meglio il valore di Cyclomatic Complexity è necessario considerare il valore di Essential Cyclomatic Complexity. Il primo parametro può essere interpretato come difficoltà di testing del metodo in quanto rappresenta il numero di percorsi di esecuzione distinti, definiti dalla presenza di punti di decisione del (if, while, for, do, switch case, catch, espressioni condizionali, and e or). Per il metodo in esame ci sono 17 possibili percorsi da testare. In genere, 1-4 è una complessità bassa, 5-7 indica una complessità moderata, 8-10 è una complessità elevata e 11+ è una complessità molto elevata.   
L’elevata complessità ciclomatica viene però compensata dal valore unitario della complessità essenziale. Questo parametro, che indica quanto sia mal strutturato il flusso di controllo di un metodo, indica che è relativamente semplice scomporre il metodo in sotto-metodi più semplici da testare.

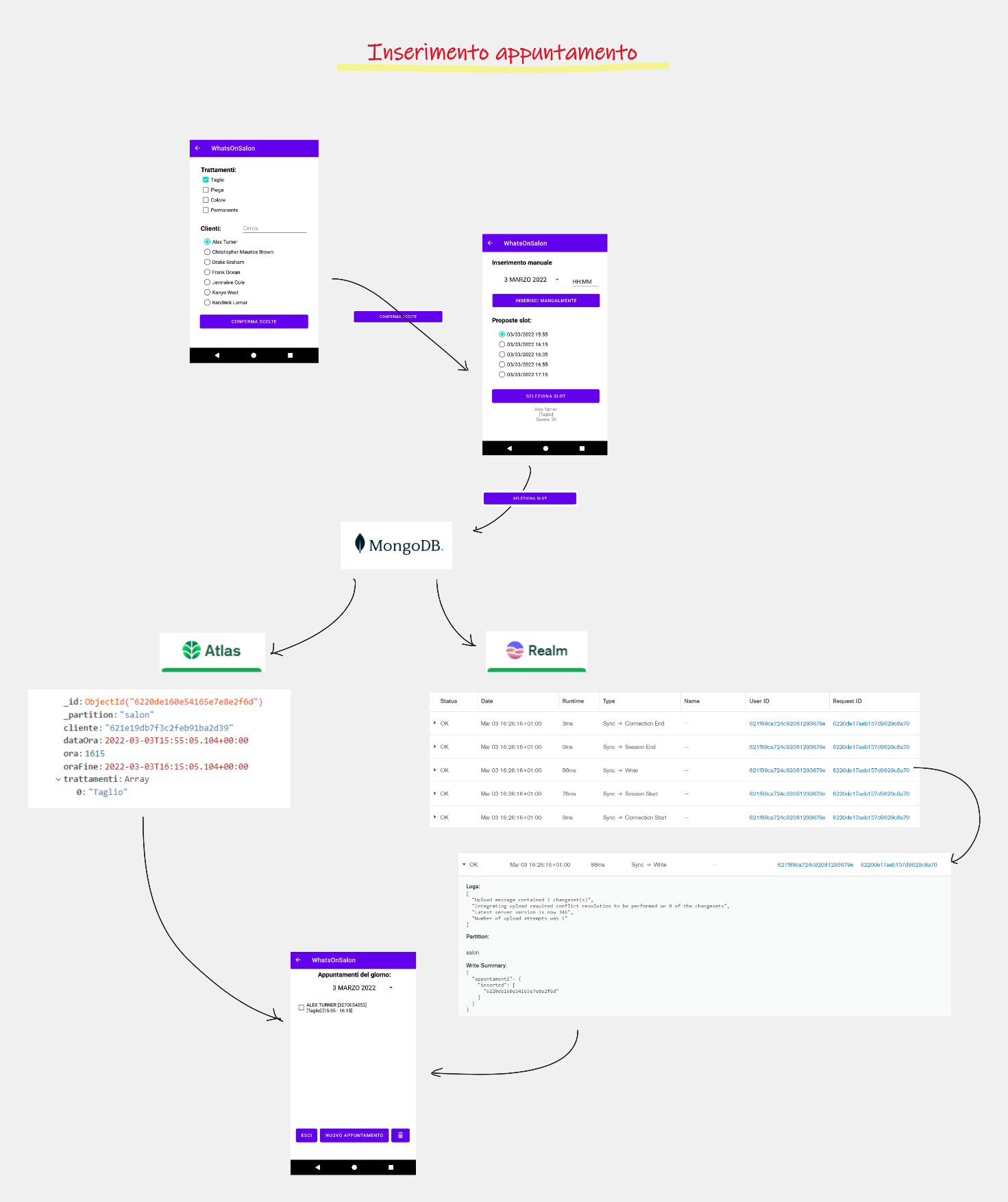
Le considerazioni fatte per questo metodo si riflettono anche per gli altri metodi con stessi risultati.

**ANALISI DINAMICA**

Per effettuare un’analisi dinamica del progetto si è preferito verificare il corretto funzionamento delle chiamate al database. Per facilitare questa fase ci si è affidati alla console di logs offerto dalla piattaforma Realm e alla verifica manuale degli inserimenti/cancellazioni dei documenti dal data lake Atlas.

A tal proposito si è simulato un inserimento di un appuntamento e alla sua successiva eliminazione dal database.

**Inserimento appuntamento**



**Cancellazione appuntamento**

