Elaborato per il corso Basi di dati

Progettazione di una base di dati per la gestione di video perizie

Buizo Manuel
Matteini Mattia
Paganelli Alberto

8 luglio 2021

Indice

Capitolo 1

Analisi dei requisiti

Si vuole realizzare un database a supporto della gestione di video perizie (perizie a distanza tramite videochiamata) effettuate per varie assicurazioni italiane.
La base di dati dovrà immagazzinare informazioni relative alle assicurazioni, ai sinistri, agli assicurati, agli studi peritali e ai relativi periti.
Tutte le assicurazioni devono avere la possibilità di controllare documenti e dati delle video-perizie effettuate.

1.1 Intervista

Si vuole tenere traccia di tutte le video-perizie effettuate, di ogni studio peritale e delle parti coinvolte.
Ogni assicurazione dovrà generare sinistri di varia natura (R.C.A., Furto, Incendio, ecc. . .).
Questi verranno assegnati agli studi, i quali si occuperanno di portare a termine le attività peritali.
Ogni studio peritale dispone di un supervisore che avrà il compito di ricevere i sinistri che arrivano dalle assicurazioni e di smistarli ai periti del proprio studio.
Il supervisore dello studio quindi creerà un incarico relativo al sinistro arrivatogli e lo assegnerà ad un perito che dovrà svolgere le attività peritali inerenti (video-perizia, richiesta documenti, ecc..).
Ogni incarico conterrà informazioni riguardanti sinistro di riferimento, perito incaricato e stato di avanzamento (Aperto, Svolgimento, Chiuso).
Inoltre includerà uno storico delle video-perizie svolte e l'insieme dei documenti richiesti all'assicurato (come eventuali contratti o documenti personali).

Il perito quando riceverà l'incarico dal supervisore, dovrà mettersi in contatto con l'assicurato al fine di definire i dettagli per l'effettuazione della video-perizia ed eventualmente richiedere dei documenti per le pratiche preliminari.
Durante una video-perizia si potranno raccogliere vari media (foto e video) che saranno allegati alla perizia.
Ogni media a sua volta è comprensivo di metadati ricavati dal GPS del dispositivo dell'assicurato.
Per ogni documento invece dovremo sapere la sua tipologia e se sarà necessaria o meno la firma dell'assicurato.
In questo modo teniamo traccia di ogni sinistro, dall'assicurazione ai tipi di documenti richiesti o alle foto georeferenziate scattate durante la video-perizia.

1.2 Definizione delle specifiche in linguaggio naturale ed estrazione dei concetti principali

1.2.1 Glossario dei termini

Termine	Descrizione	Sinonimo
Assicurazione	Colei che riceve da cittadini nuove richieste e crea sinistri	Ente esterno
Studio	Colei che riceve da cittadini nuove richieste e crea sinistri	Ente esterno
Supervisore	Colui che, all'interno dello studio, ha il compito di generare incarichi e assegnarli ad uno dei propri periti	Coordinatore
Perito	Colui che si occupa della attività peritali	Incaricato dal supervisore, membro dello studio/ufficio
Assicurato	Colui che farà parte alla videoperizia e che si occupa della ripresa del sinistro	Cliente, parte coinvolta
Sinistro	Danno e relativo tipo di danno da periziare	
Incarico	Insieme di attività peritali atte all'intero svolgimento della perizia	Fascicolo
Video-perizia	Perizia eseguita telematicamente tramite smartphone o dispositivo mobile	Perizia telematica

Documenti	Documenti richiesti al fine di eseguire una perizia completa	
Media	Media raccolti durante la video-perizia, comprensivi di metadati	Foto, video
Metadati	Informazioni raccolte dal dispositivo dell'assicurato, in questo caso specifico quelli relativi alla geolocalizzazione	

1.2.2 Riassunto dei concetti principali

Ogni Sinistro, è individuato tramite un identificativo incrementale e può essere di un solo `TIPO_SINISTRO`.
Un' Assicurazione, identificata tramite la sua denominazione, cede la gestione del `SINISTRO` allo `STUDIO` (identificato dalla P.Iva e da un identificativo incrementale) e lo stesso sinistro non può essere assegnato ad un altro ufficio.
Ogni studio può avere più di un `SUPERVISORE` ma ne ha almeno uno.
Gli `STUDI` possono ricevere sinistri da tutte le `ASSICURAZIONI` e un `SUPERVISORE` deve creare un `INCARICO` e assegnarne la gestione ad un proprio `PERITO`.
Non può essere assegnato un `INCARICO` a più di un `PERITO` ma a un `PERITO` possono essere assegnati più `INCARICHI` e da più `SUPERVISORI`.
Il `PERITO`, che avrà il compito di svolgere le attività peritali inerenti all'`INCARICO` a lui assegnato, dovrà poter svolgere anche più di una `VIDEO-PERIZIA` per entrare in contatto con l'`ASSICURATO` e poter scrivere una descrizione del danno (ad esempio se si vede meglio in altra fase della giornata, danno grande che richiede più videochiamate, ecc...).
Potrà anche lavorare a più `INCARICHI` alla volta e nello stesso giorno.
L'`ASSICURATO` invece sarà registrato tramite un'anagrafica ed identificato mediante il codice fiscale.
La `VIDEO-PERIZIA` e i `MEDIA` raccolti, sono relativi esclusivamente ad un `INCARICO`.
Ogni `INCARICO` possiede anche una raccolta di `DOCUMENTI` riguardanti il sinistro.
I `DOCUMENTI` riguardano principalmente l'assicurato e il tipo di sinistro, non sapendo quindi quanti documenti possono essere richiesti, non vi è nessun vincolo.

Ogni `VIDEO-PERIZIA` deve comprendere anche un luogo effettivo e sarà quindi localizzato tramite coordinate e sistema di riferimento. (Altitudine, Latitudine, Longitudine e Est o Ovest).
Ogni `VIDEO-PERIZIA` ed ogni `MEDIA` deve essere localizzato per essere definita valida.

Segue un elenco delle principali azioni richieste:

1. Registrare/eliminare una nuova assicurazione aderente
2. Registrare/eliminare studi peritali
3. Registrare/eliminare periti per uno studio
4. se la soluzione è stata realizzata utilizzando uno
5. Registrare/cambiare supervisori per uno studio
6. Registrare/eliminare un assicurato tramite la sua anagrafica
7. Assegnare il perito all'incarico
8. Gestire lo stato degli incarichi
9. Leggere tutti gli incarichi aperti per una data assicurazione o studio
10. Registrare/eliminare documenti per un incarico
11. Inserire una video-perizia per un incarico
12. Inserire in database tutti i media raccolti in fase di video-perizia
13. Tenere traccia di ogni sinistro avvenuto in una fascia temporale
14. Avere una lista di ogni video-perizia svolta per ogni assicurazione o studio

Capitolo 2

Progettazione Concettuale

In questo capitolo si spiegano le strategie messe in campo per soddisfare i requisiti identificati nell'analisi.

2.1 Schema scheletro

Questa sezione spiega come le componenti principali del software interagiscono fra loro. In particolare, qui va spiegato **se** e **come** è stato utilizzato il pattern architetturale model-view-controller (e/o alcune sue declinazioni specifiche, come entity-control-boundary). In questa sezione vanno descritte, per ciascun componente architetturale che ruoli ricopre (due o tre ruoli al massimo), ed in che modo interagisce (ossia, scambia informazioni) con gli altri componenti dell'architettura. Raccomandiamo di porre particolare attenzione al design dell'interazione fra view e controller: se ben progettato, sostituire in blocco la view non dovrebbe causare alcuna modifica nel controller (tantomeno nel model).

Con questa architettura, possono essere aggiunti un numero arbitrario di input ed output all'intelligenza artificiale. Ovviamente, mentre l'aggiunta di output è semplice e non richiede alcuna modifica all'IA, la presenza di nuovi tipi di evento richiede invece in potenza aggiunte o rifiniture a GLaDOS. Questo è dovuto al fatto che nuovi Input rappresentano di fatto nuovi elementi della business logic, la cui alterazione od espansione inevitabilmente impatta il controller del progetto.

In **??** è esemplificato il diagramma UML architetturale.

2.2 Raffinamenti proposti

È assolutamente inutile, ed è anzi controproducente, descrivere classe-per-classe (o peggio ancora metodo-per-metodo) com'è fatto il vostro software: è un livello di dettaglio proprio della documentazione dell'API (deducibile dalla Javadoc).

È necessario che ciascun membro del gruppo abbia una propria sezione di design dettagliato, di cui sarà il solo responsabile. Per continuare il parallelo con la vettura di Formula 1, se nei fogli di progetto che mostrano il design delle sospensioni anteriori appaiono pezzi che appartengono al volante o al turbo, c'è una chiara indicazione di qualche problema di design.

Si divida la sezione in sottosezioni, e per ogni aspetto di design che si vuole approfondire, si presenti:

1. : una breve descrizione in linguaggio naturale del problema che si vuole risolvere, se necessario ci si può aiutare con schemi o immagini;

2. : una descrizione della soluzione proposta, analizzando eventuali alternative che sono state prese in considerazione, e che descriva pro e contro della scelta fatta;

3. : uno schema UML che aiuti a comprendere la soluzione sopra descritta;

4. : se la soluzione è stata realizzata utilizzando uno o più pattern noti, si spieghi come questi sono reificati nel progetto (ad esempio: nel caso di Template Method, qual è il metodo template; nel caso di Strategy, quale interfaccia del progetto rappresenta la strategia, e quali sono le sue implementazioni; nel caso di Decorator, qual è la classe astratta che fa da Decorator e quali sono le sue implementazioni concrete; eccetera);

La presenza di pattern di progettazione *correttamente utilizzati* è valutata molto positivamente. L'uso inappropriato è invece valutato negativamente: a tal proposito, si raccomanda di porre particolare attenzione all'abuso di Singleton, che, se usato in modo inappropriato, è di fatto un anti-pattern.

2.3 Schema concettuale finale

Problema GLaDOS ha più personalità intercambiabili, la cui presenza deve essere trasparente al client.

Soluzione Il sistema per la gestione della personalità utilizza il *pattern Strategy*, come da **??**: le implementazioni di `Personality` possono essere modificate, e la modifica impatta direttamente sul comportamento di GLaDOS.

Problema In fase di sviluppo, sono state sviluppate due personalità, una buona ed una cattiva. Quella buona restituisce sempre una torta vera, mentre quella cattiva restituisce sempre la promessa di una torta che verrà in realtà disattesa. Ci si è accorti che diverse personalità condividevano molto del comportamento, portando a classi molto simili e a duplicazione.

Soluzione Dato che le due personalità differiscono solo per il comportamento da effettuarsi in caso di percorso completato con successo, è stato utilizzato il *pattern template method* per massimizzare il riuso, come da **??**. Il metodo template è `onSuccess()`, che chiama un metodo astratto e protetto `makeCake()`.

Gestione di output multipli

Figura 2.1: Il pattern Observer è usato per consentire a GLaDOS di informare tutti i sistemi di output in ascolto

Problema Il sistema deve supportare output multipli. In particolare, si richiede che vi sia un logger che stampa a terminale o su file, e un'interfaccia grafica che mostri una rappresentazione grafica del sistema.

Soluzione Dato che i due sistemi di reporting utilizzano le medesime informazioni, si è deciso di raggrupparli dietro l'interfaccia `Output`. A questo punto, le due possibilità erano quelle di far sì che `GLaDOS` potesse pilotarle entrambe. Invece di fare un sistema in cui questi output sono obbligatori e connessi, si è deciso di usare maggior flessibilità (anche in vista di future estensioni) e di adottare una comunicazione uno-a-molti fra `GLaDOS` ed i sistemi di output. La scelta è quindi ricaduta sul *pattern Observer*: `GLaDOS` è observable, e le istanze di `Output` sono observer. Il suo utilizzo è esemplificato in Figura 2.1

Contro-esempio: pessimo diagramma UML

In Figura 2.2 è mostrato il modo **sbagliato** di fare le cose. Questo schema è fatto male perché:

- È caotico.
- È difficile da leggere e capire.
- Vi sono troppe classi, e non si capisce bene quali siano i rapporti che intercorrono fra loro.
- Si mostrano elementi implementativi irrilevanti, come i campi e i metodi privati nella classe `AbstractEnvironment`.
- Se l'intenzione era quella di costruire un diagramma architetturale, allora lo schema è ancora più sbagliato, perché mostra pezzi di implementazione.
- Una delle classi, in alto al centro, galleggia nello schema, non connessa a nessuna altra classe, e di fatto costituisce da sola un secondo schema UML scorrelato al resto
- Le interfacce presentano tutti i metodi e non una selezione che aiuti il lettore a capire quale parte del sistema si vuol mostrare.

Figura 2.2: Schema UML mal fatto e con una pessima descrizione, che non aiuta a capire. Don't try this at home.

Capitolo 3

Progettazione Logica

3.1 Stima del volume dei dati

Il testing automatizzato è un requisito di qualunque progetto software che si rispetti, e consente di verificare che non vi siano regressioni nelle funzionalità a fronte di aggiornamenti. Per quanto riguarda questo progetto è considerato sufficiente un test minimale, a patto che sia completamente automatico. Test che richiedono l'intervento da parte dell'utente sono considerati *negativamente* nel computo del punteggio finale.

3.2 Descrizione delle operazioni principali e stima della loro frequenza

Ci aspettiamo, leggendo questa sezione, di trovare conferma alla divisione operata nella sezione del design di dettaglio, e di capire come è stato svolto il lavoro di integrazione. **Andrà realizzata una sotto-sezione separata per ciascuno studente** che identifichi le porzioni di progetto sviluppate, separando quelle svolte in autonomia da quelle sviluppate in collaborazione. Diversamente dalla sezione di design, in questa è consentito elencare package/classi, se lo studente ritiene sia il modo più efficace di convogliare l'informazione. Si ricorda che l'impegno deve giustificare circa 40-50 ore di sviluppo (è normale e fisiologico che approssimativamente la metà del tempo sia impiegata in analisi e progettazione).

3.3 Schemi di navigazione e tabelle degli accessi

Questa sezione, come quella riguardante il design dettagliato va svolta **singolarmente da ogni membro del gruppo**.

In questa sezione, *dopo l'elenco*, è anche bene evidenziare eventuali pezzi di codice "riadattati" (o scopiazzati...) da Internet o da altri progetti, pratica che tolleriamo ma che non raccomandiamo.

3.4 Raffinamento dello schema

Questa sezione, come quella riguardante il design dettagliato va svolta **singolarmente da ogni membro del gruppo**.

3.5 Analisi delle ridondanze

Questa sezione, come quella riguardante il design dettagliato va svolta **singolarmente da ogni membro del gruppo**.

3.6 Traduzione di entità e associazioni in relazioni

Questa sezione, come quella riguardante il design dettagliato va svolta **singolarmente da ogni membro del gruppo**.

3.7 Schema relazionale finale

Questa sezione, come quella riguardante il design dettagliato va svolta **singolarmente da ogni membro del gruppo**.

3.8 Traduzione delle operazioni in query SQL

Questa sezione, come quella riguardante il design dettagliato va svolta **singolarmente da ogni membro del gruppo**.

Capitolo 4

Progettazione dell'applicazione

In quest'ultimo capitolo si tirano le somme del lavoro svolto e si delineano eventuali sviluppi futuri.

Nessuna delle informazioni incluse in questo capitolo verrà utilizzata per formulare la valutazione finale, a meno che non sia assente o manchino delle sezioni obbligatorie. Al fine di evitare pregiudizi involontari, l'intero capitolo verrà letto dai docenti solo dopo aver formulato la valutazione.

4.1 Autovalutazione e lavori futuri

È richiesta una sezione per ciascun membro del gruppo, obbligatoriamente. Ciascuno dovrà autovalutare il proprio lavoro, elencando i punti di forza e di debolezza in quanto prodotto. Si dovrà anche cercare di descrivere *in modo quanto più obiettivo possibile* il proprio ruolo all'interno del gruppo. Si ricorda, a tal proposito, che ciascuno studente è responsabile solo della propria sezione: non è un problema se ci sono opinioni contrastanti, a patto che rispecchino effettivamente l'opinione di chi le scrive. Nel caso in cui si pensasse di portare avanti il progetto, ad esempio perché effettivamente impiegato, o perché sufficientemente ben riuscito da poter esser usato come dimostrazione di esser capaci progettisti, si descriva brevemente verso che direzione portarlo.

4.2 Difficoltà incontrate e commenti per i docenti

Questa sezione, **opzionale**, può essere utilizzata per segnalare ai docenti eventuali problemi o difficoltà incontrate nel corso o nello svolgimento del

progetto, può essere vista come una seconda possibilità di valutare il corso (dopo quella offerta dalle rilevazioni della didattica) avendo anche conoscenza delle modalità e delle difficoltà collegate all'esame, cosa impossibile da fare usando le valutazioni in aula per ovvie ragioni. È possibile che alcuni dei commenti forniti vengano utilizzati per migliorare il corso in futuro: sebbene non andrà a vostro beneficio, potreste fare un favore ai vostri futuri colleghi. Ovviamente *il contenuto della sezione non impatterà il voto finale.*