Brew Day documentation

Martino Andrian 845417, Gioele De Pianto 845002, Anselmo Folcio 844827

Gennaio 2021

Contents

1 Introduzione

L'applicazione Brew Day offre un servizio di organizzazione delle attività di homebrewing permettendo al cliente di memorizzare le proprie ricette, tenere nota dei risultati ottenuti per ogni lotto prodotto e gestire il magazzino degli ingredienti.

2 Analisi dei requisiti

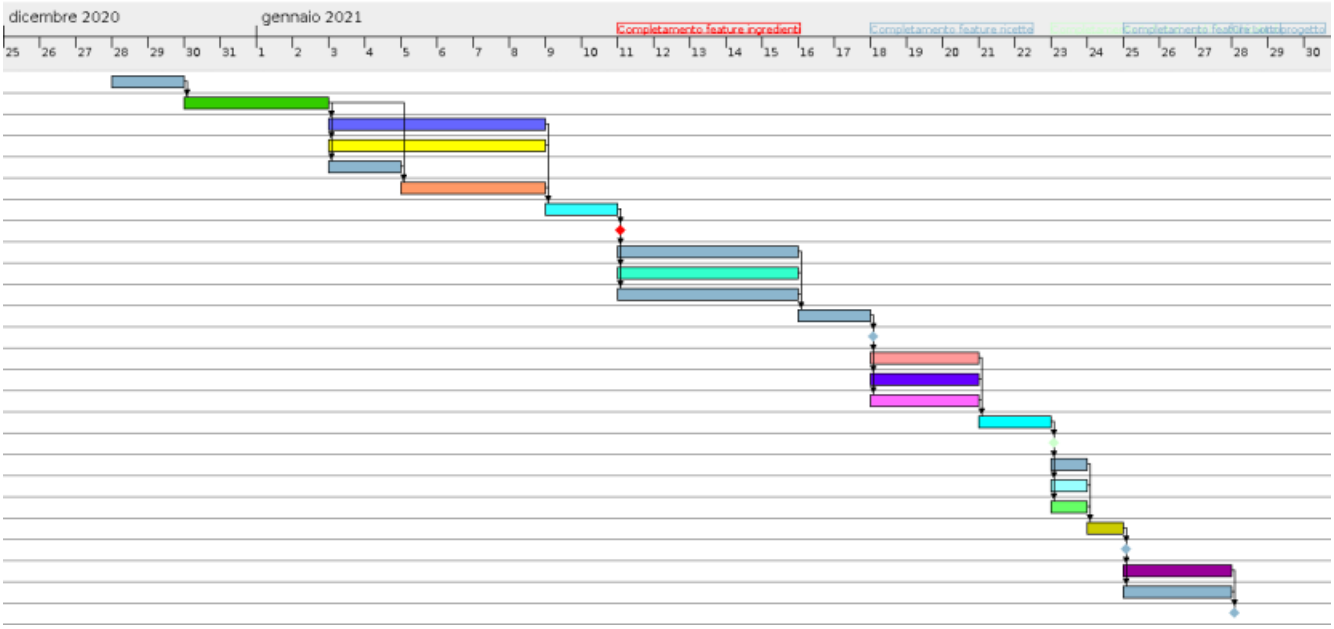
ID	Descrizione	Tipo	Priorità
1	L'utente deve poter registrare una propria password personale al primo avvio dell'applicazione	Autenticazione Funzionale	M
2	L'utente deve poter effettuare il login specificando la propria password ad ogni avvio dell'applicazione successivo al primo	Autenticazione Funzionale	M
3	Il sistema dovrà permettere di visualizzare una lista degli ingredienti disponibili inseriti dall'utente, dovrà inoltre permettergli di eliminare, modificare o aggiornare tali ingredienti	Ingredienti Funzionale	M
4	Il sistema Brew Day dovrà permettere all'utente di visualizzare nella home la lista di ricette inserite dall'utente permettendogli di eliminarle o modificarle, dovrà inoltre permettergli di inserire nuove ricette attraverso un'apposita schermata.	Ricette Funzionale	M
5	All'avvio dell'applicazione per la prima volta, dopo la registrazione, il sistema deve obbligatoriamente permettere all'utente di inserire il proprio equipaggiamento prima di compiere ulteriori azioni	Equipaggiamento Funzionale	M
6	Il sistema Brew Day deve notificare la scarsità degli ingredienti attraverso un'apposita vista	Lista della Spesa Funzionale	S
7	Il sistema Brew Day deve permettere all'utente di modificare il proprio equipaggiamento, specificando la sua nuova capacità produttiva	Equipaggiamento Funzionale	M
8	Ad ogni ciclo produttivo il sistema Brew Day deve aggiornare le disponibilità degli ingredienti	Lista della Spesa Funzionale	M
9	Ogni volta che viene prodotta una ricetta il sistema Brew Day deve permettere all'utente di aggiungere una nota sul lotto prodotto, eventualmente dandone anche una valutazione	Nota Funzionale	S

3 Diagramma Gannt e organizzazione del lavoro

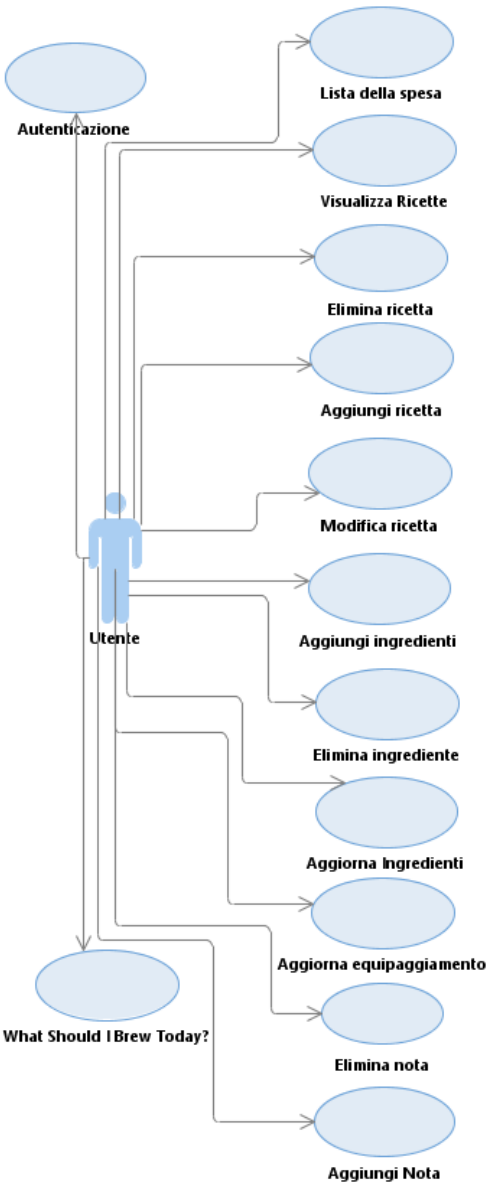
3.1 Descrizione delle attività

Nome	Data d'inizio	Data di fine
Inizio progetto e analisi tempi	28/12/20	29/12/20
Analisi e progettazione	30/12/20	02/01/21
Analisi tecnologia DB	03/01/21	04/01/21
Implementazione interfaccia grafica per feature ingredienti	03/01/21	08/01/21
Implementazione backend feature ingredienti	03/01/21	08/01/21
Implementazione Database ingredienti	05/01/21	08/01/21
Documentazione analisi sonarqube e miglioramento codice	09/01/21	10/01/21
Completamento feature ingredienti	11/01/21	11/01/21
Implementazione interfaccia grafica Ricette	11/01/21	15/01/21
Implementazione backend feature ricette	11/01/21	15/01/21
Database feature ricette	11/01/21	15/01/21
Documentazione, analisi sonarqube e miglioramento codice	16/01/21	17/01/21
Completamento feature ricette	18/01/21	18/01/21
Implementazione interfaccia grafica feature "what should I brew today?" + lista della spesa	18/01/21	20/01/21
Implementazione backend feature "what should I brew today?" + lista della spesa	18/01/21	20/01/21
Database "what should I brew today?" + lista della spesa	18/01/21	20/01/21
Documentazione e supporto feature "what should I brew today?", spesa e analisi sonarqube e miglioramento codice	21/01/21	22/01/21
Completamento "what should I brew today?" + lista spesa	23/01/21	23/01/21
Implementazione interfaccia grafica feature Lotto	23/01/21	23/01/21
implementazione backend feauture lotto	23/01/21	23/01/21
implementazione database feature Lotto	23/01/21	23/01/21
Documentazione, supporto, analisi sonarqube	24/01/21	24/01/21
Completamento feature Lotto	25/01/21	25/01/21
Completamento documentazione	25/01/21	27/01/21
Packaging	25/01/21	27/01/21
Chiusura progetto	28/01/21	28/01/21

3.2 Calendario delle attività



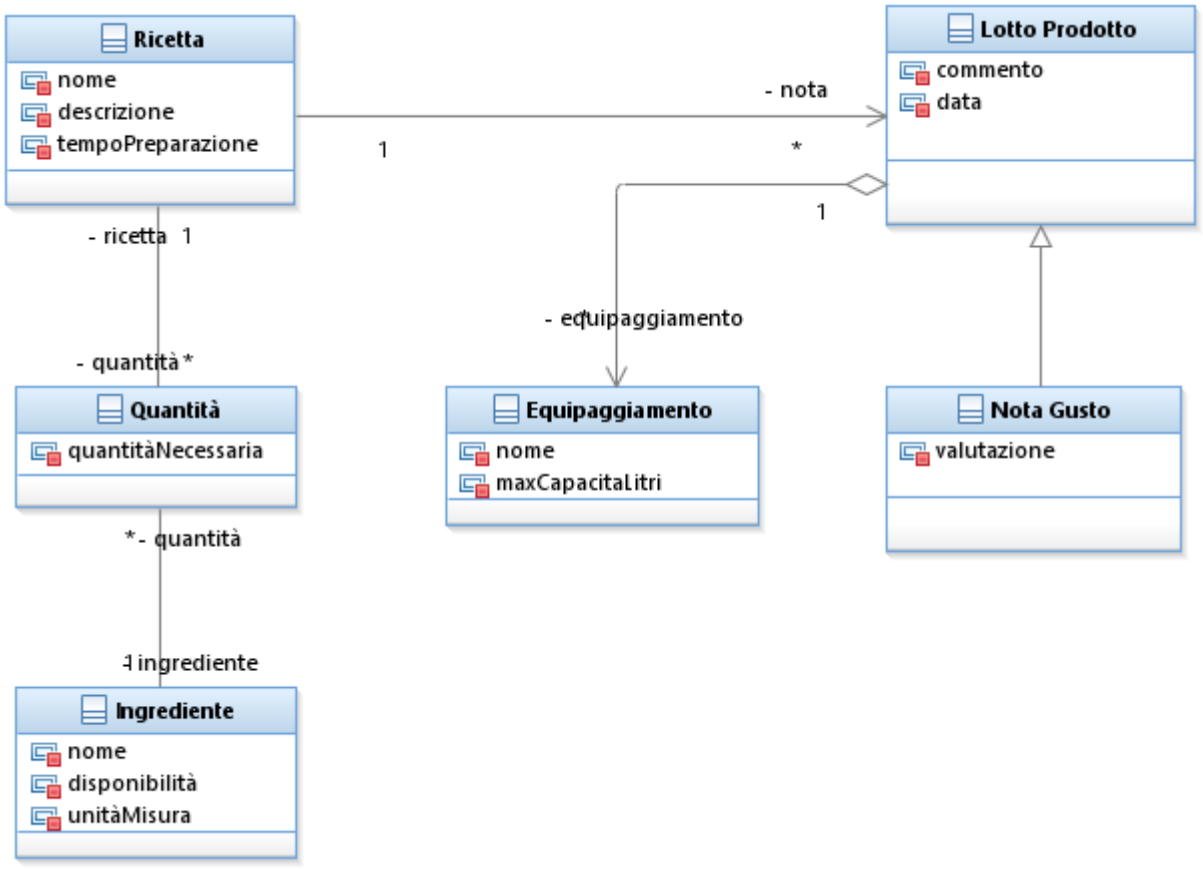
4 Diagramma dei casi D'uso



Brew Day è un'applicazione di discrete dimensioni, per semplicità e chiarezza sono stati definiti dei casi d'uso non eccessivamente grandi. Sono stati definiti in media 3 casi d'uso per ogni sfera di funzionalità del sistema (Ricetta, Ingrediente, Nota, Autenticazione, Equipaggiamento).

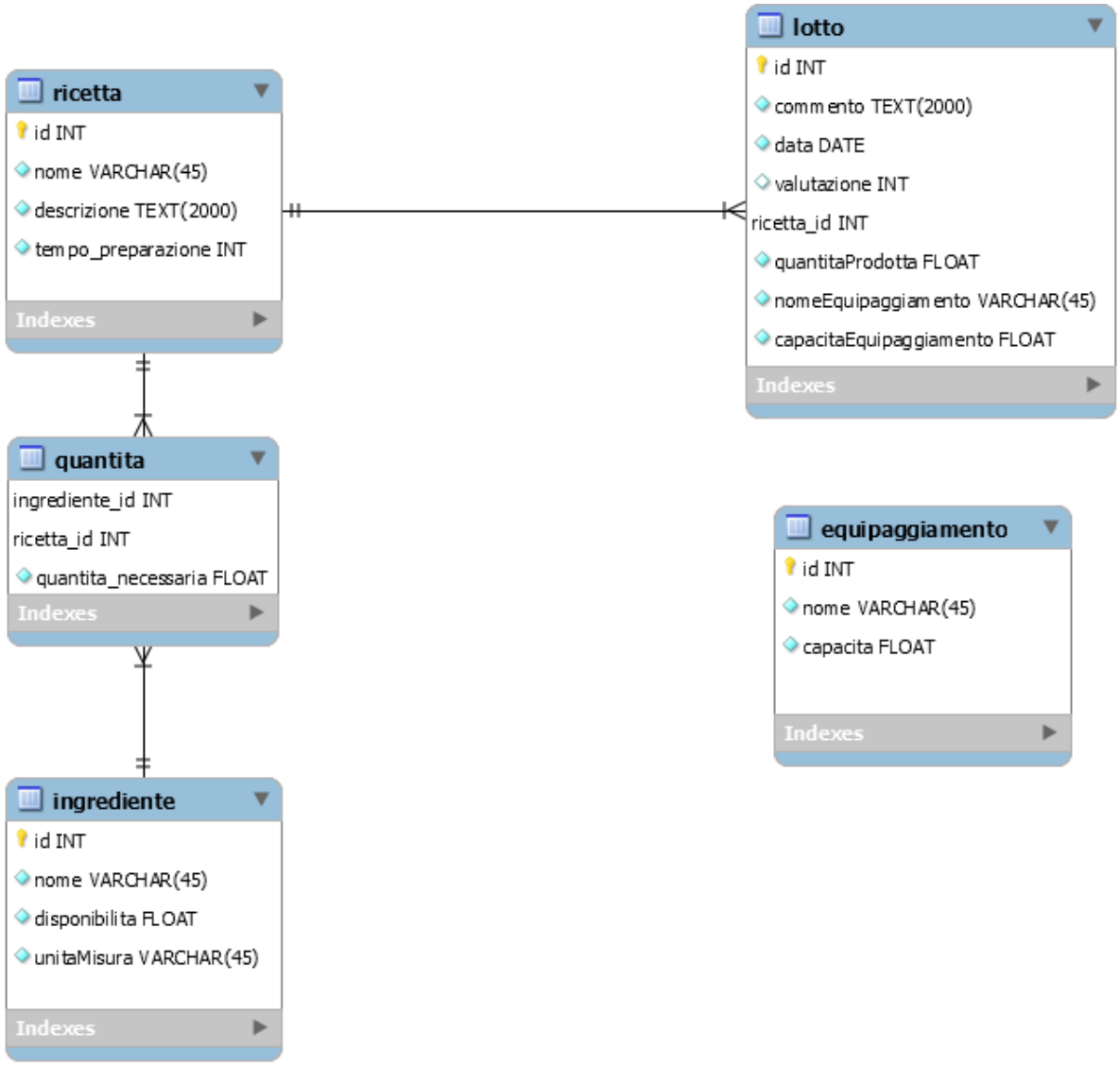
5 Diagrammi di dominio

5.1 Diagrammma delle classi di dominio



Il diagramma delle classi a livello di dominio è stato uno dei primo diagrammi sviluppati per definire con chiarezza gli oggetti concreti su cui il sistema BrewDay opera.

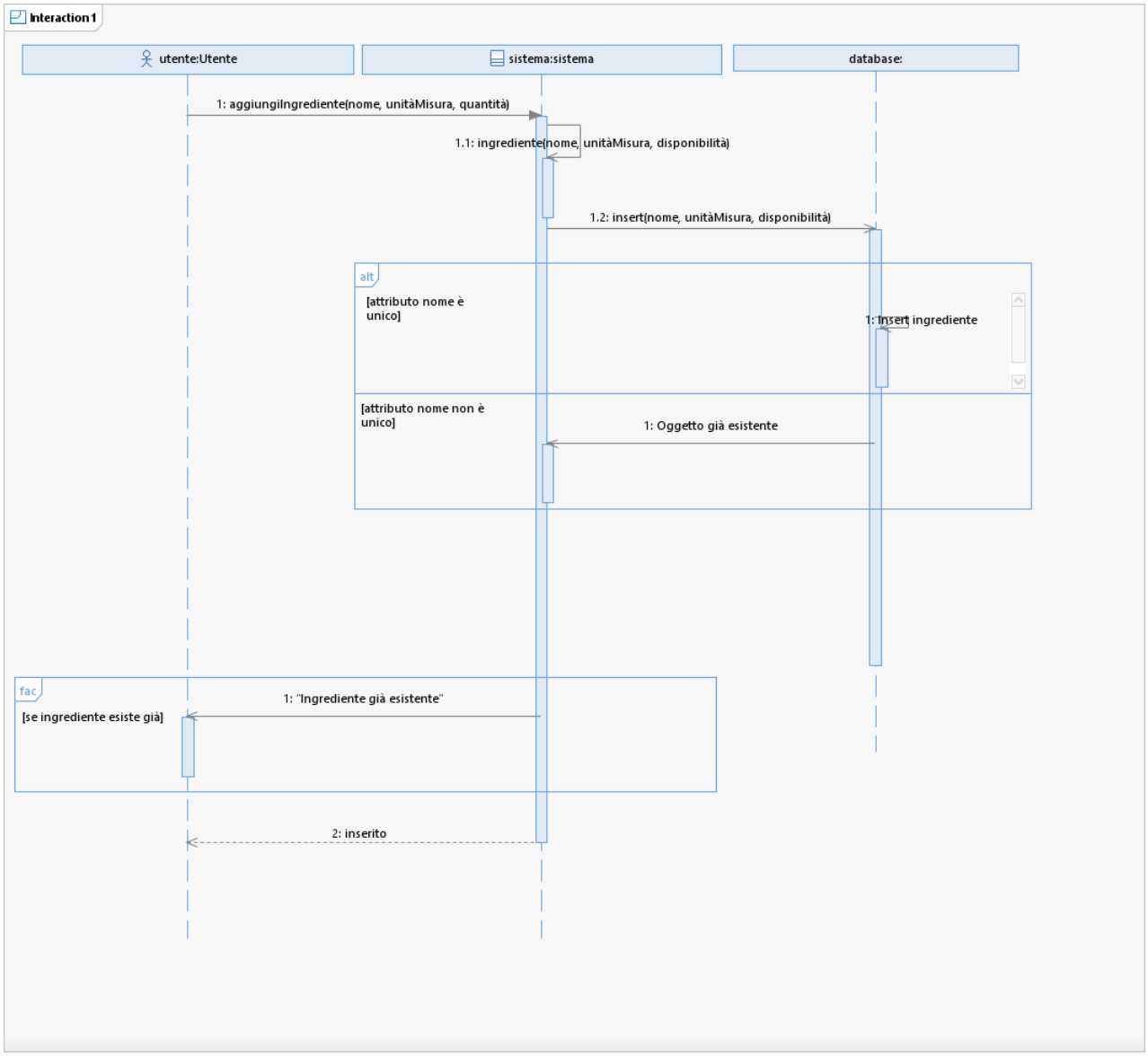
5.2 Diagramma EER



Molto simile al diagramma delle classi di dominio, il diagramma EER è utile per definire le tabelle che sono state aggiunte sucessivamente nel database e le loro relazioni.

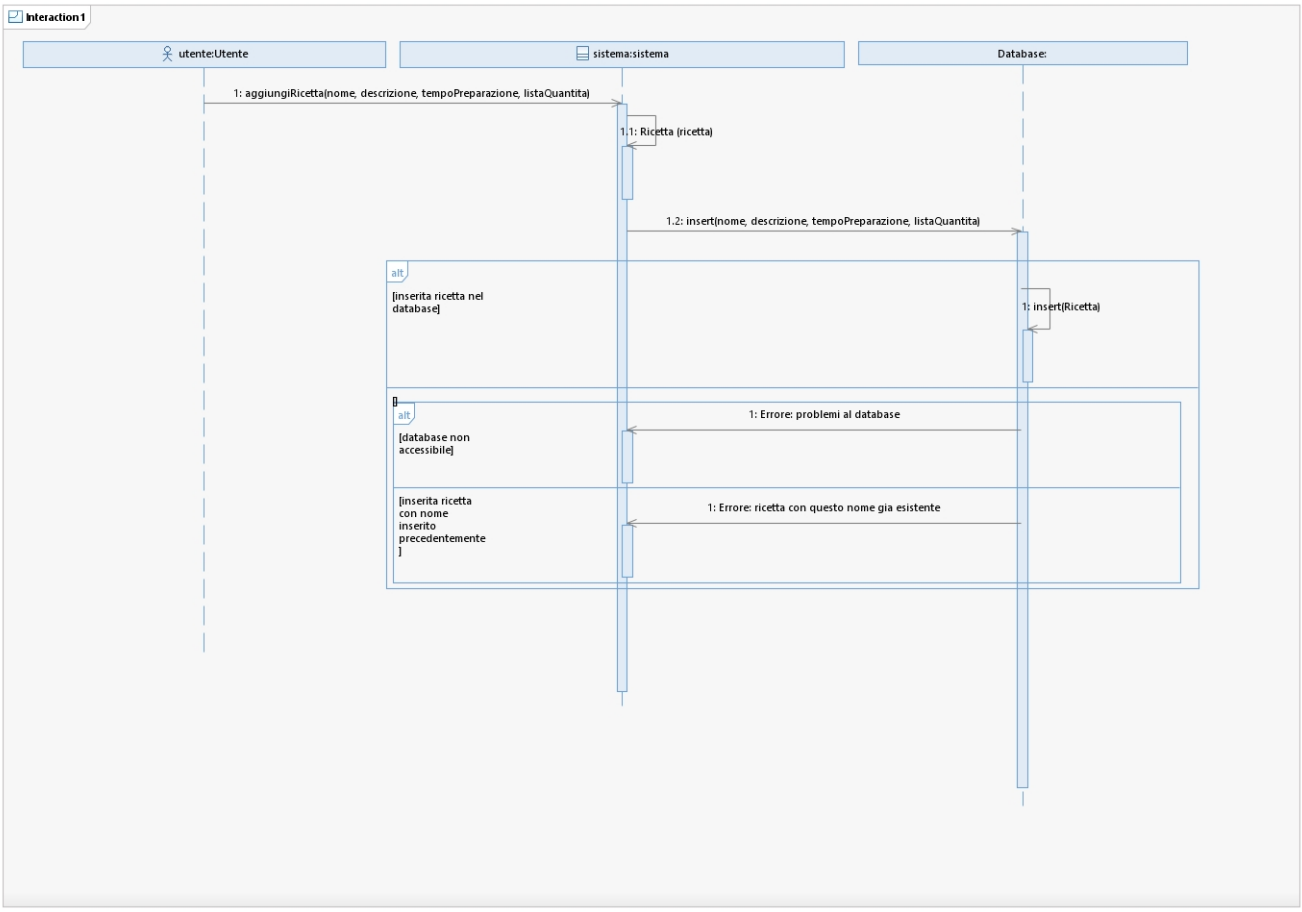
6 Diagrammi SSD

6.1 SSD aggiunta ingrediente



L'SSD dell'operazione aggiungi ingrediente illustra come, una volta invocato il metodo, il sistema crei l'oggetto che mantiene in memoria e interroga il database affinchè memorizzi l'ingrediente solo se non vi è una copia con lo stesso nome.

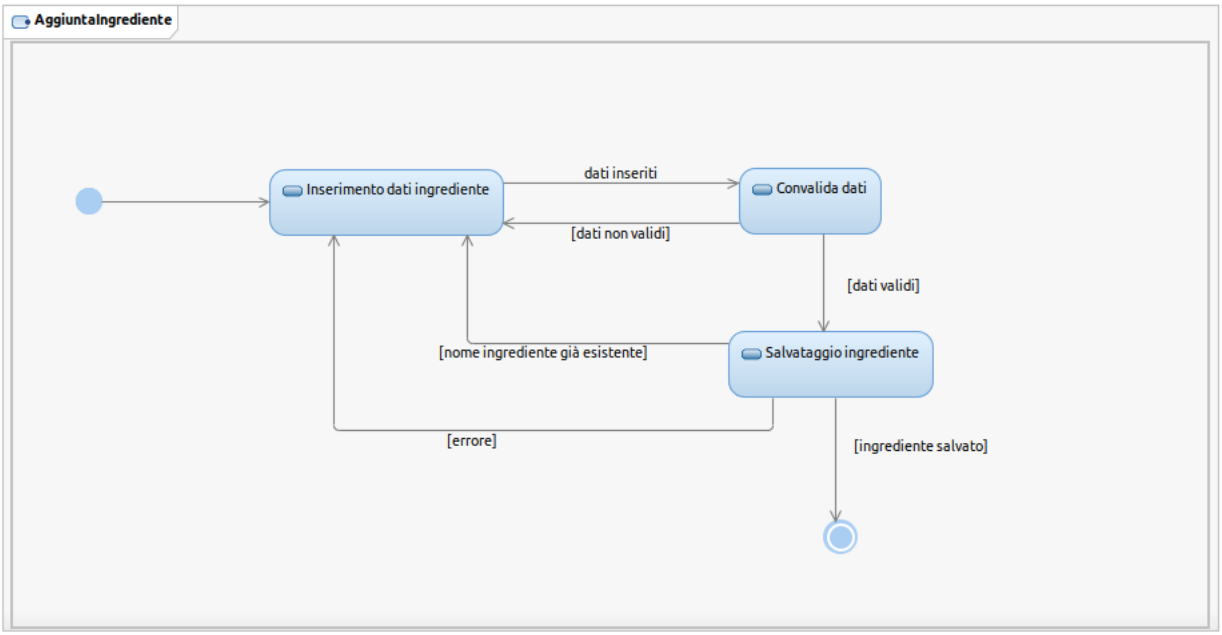
6.2 SSD aggiunta ricetta



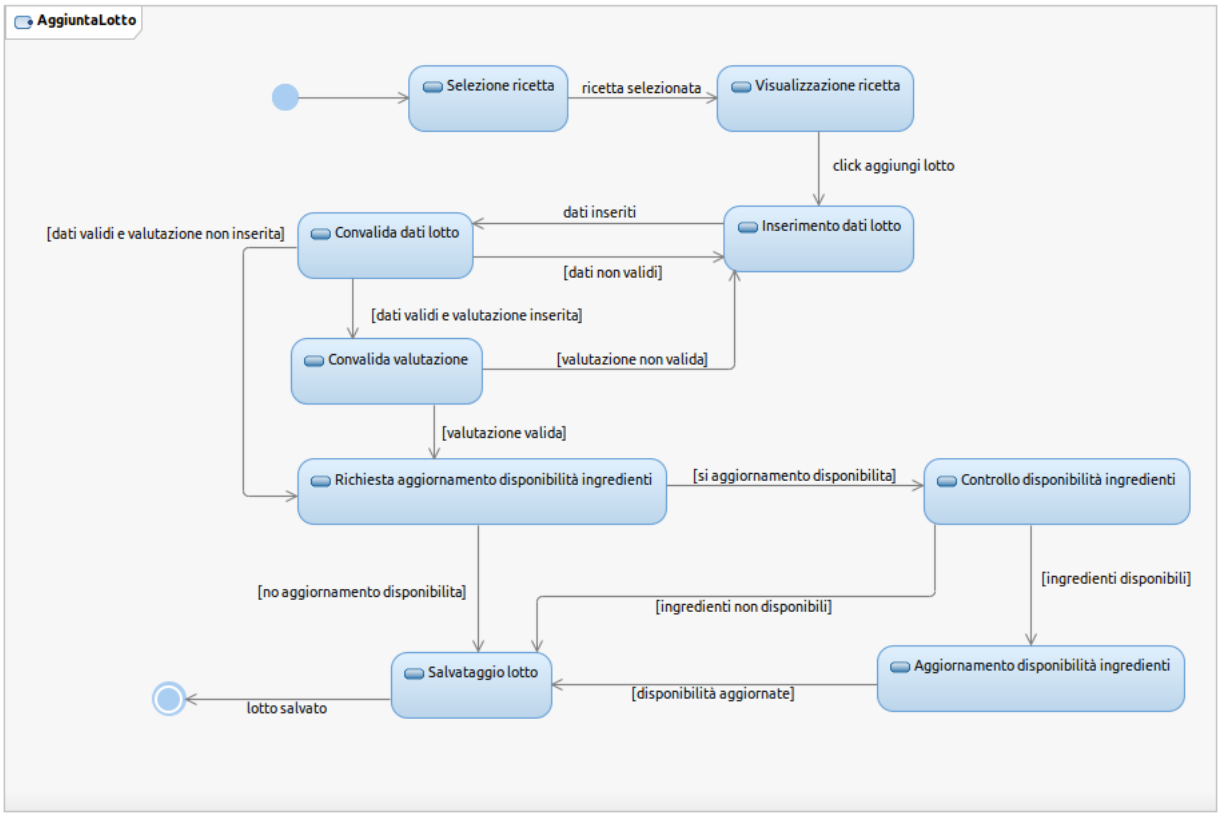
Analogamente all'aggiunta degli ingredienti l'SSD di aggiunta ricetta mostra come il sistema, una volta invocato il metodo crei in locale l'oggetto Ricetta e successivamente interroghi il database per scoprire se tale oggetto è già registrato, se non esiste un oggetto con lo stesso nome allora viene creata una nuova ricetta, altrimenti il sistema segnala l'incongruenza.

7 Diagrammi di stato

7.1 Diagramma di stato Aggiungi ricetta

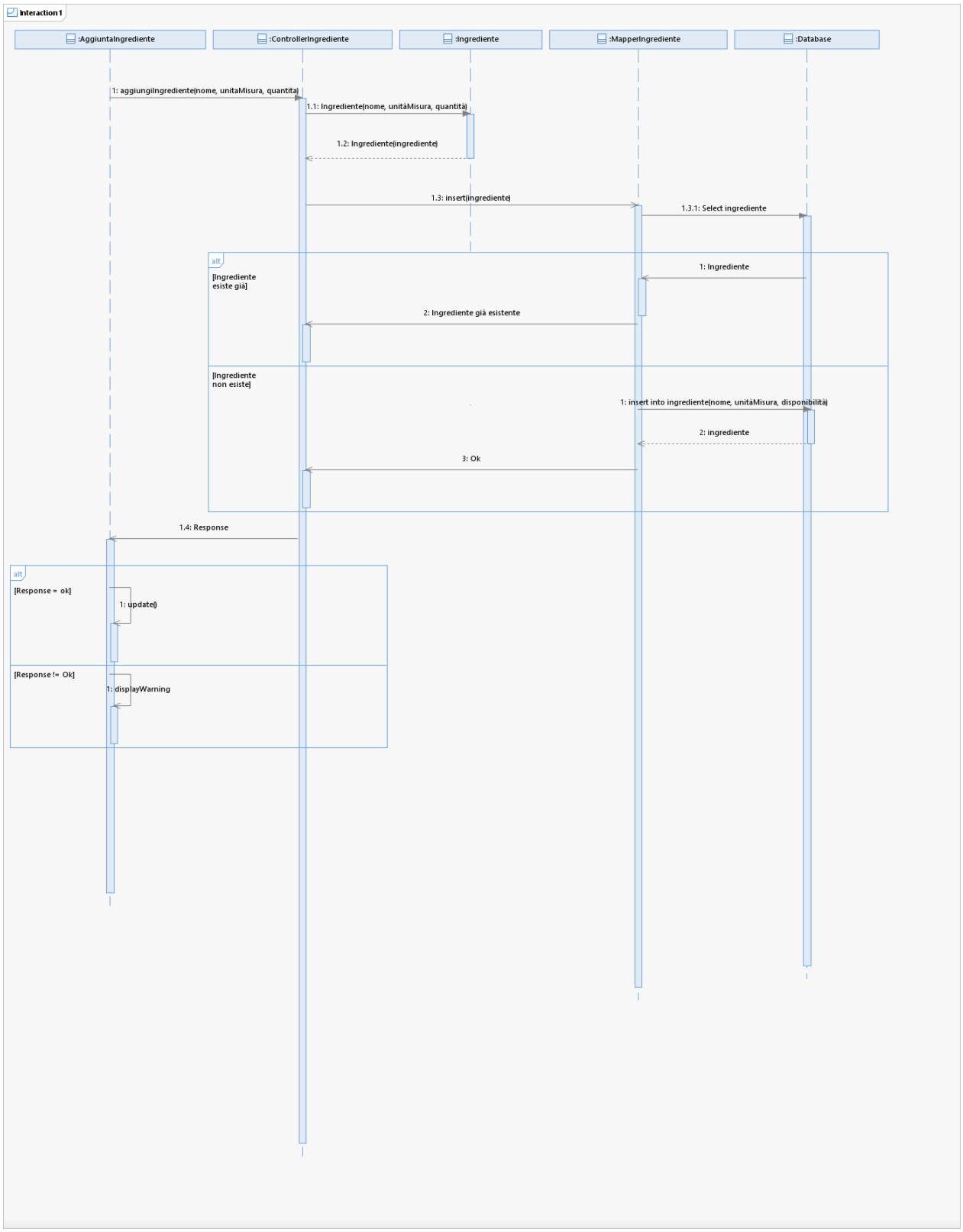


7.2 Diagramma di stato Aggiungi lotto



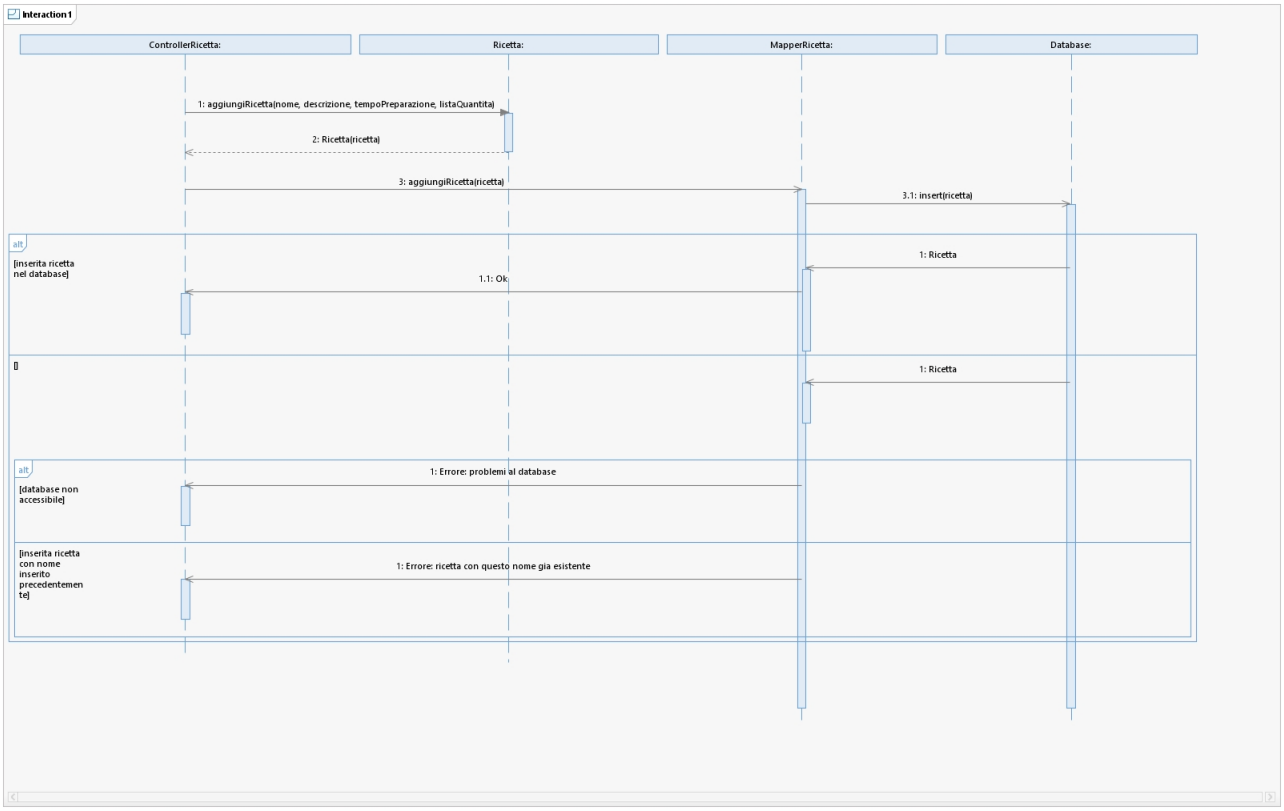
8 Diagrammi di sequenza di progettazione

8.1 Diagramma di sequenza aggiunta ingredienti



Il diagramma di sequenza aggiunta ingredienti mostra che una volta invocato il metodo del controller aggiungiIngrediente il model crea un oggetto locale mentre il mapper comunica col database per scoprire se l'oggetto è già memorizzato, se vengono superati i controlli la vista viene aggiornata con il nuovo ingrediente, altrimenti viene visualizzato un warning.

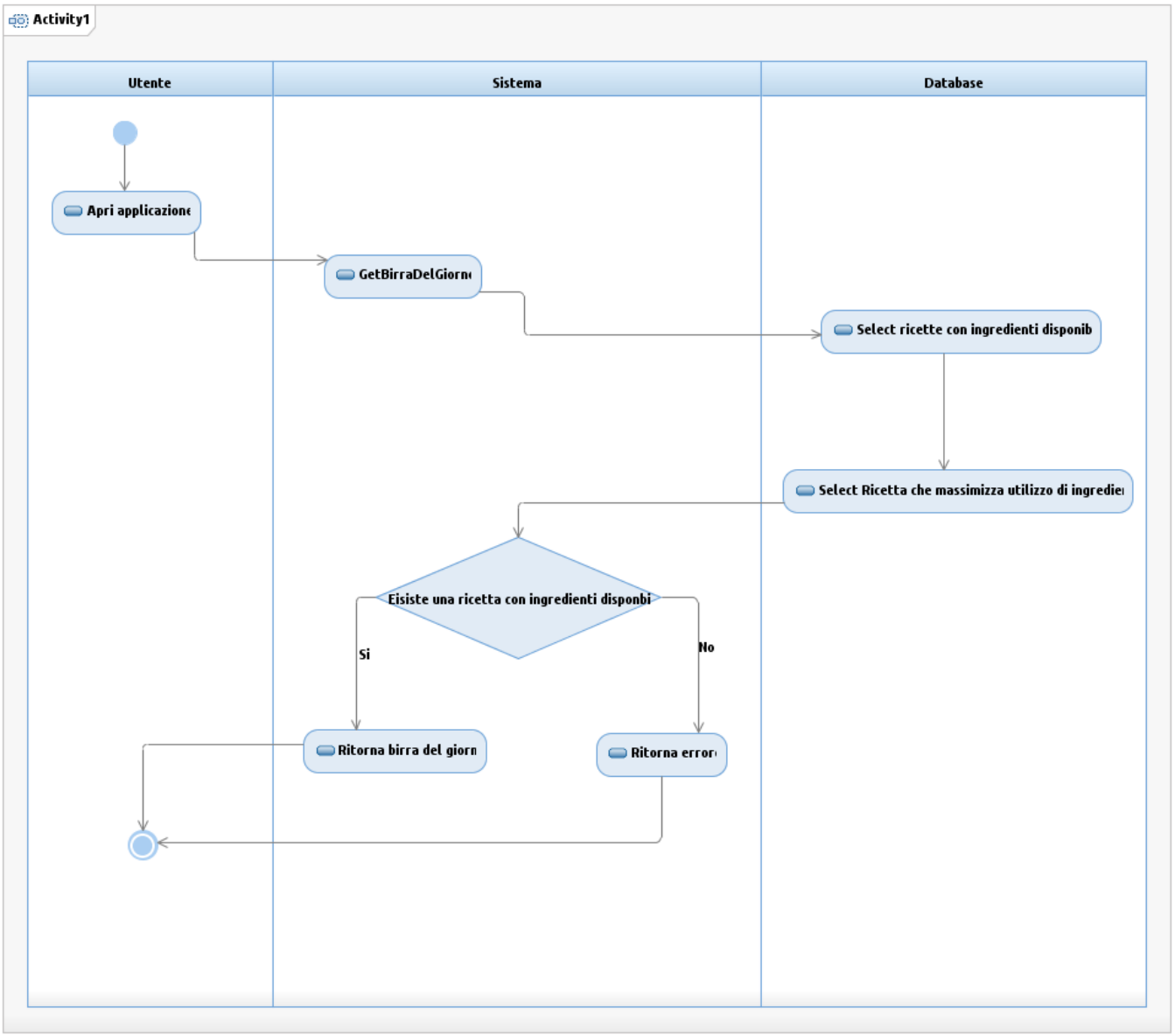
8.2 Diagramma di sequenza aggiunta ricette



Il diagramma di sequenza aggiunta ricette illustra come in modo analogo agli ingredienti il controller chiama il model per creare l'oggetto ricetta, mentre il mapper comunica con il database per stabilire se i controlli sono superati ed eventualmente aggiungere la ricetta al database.

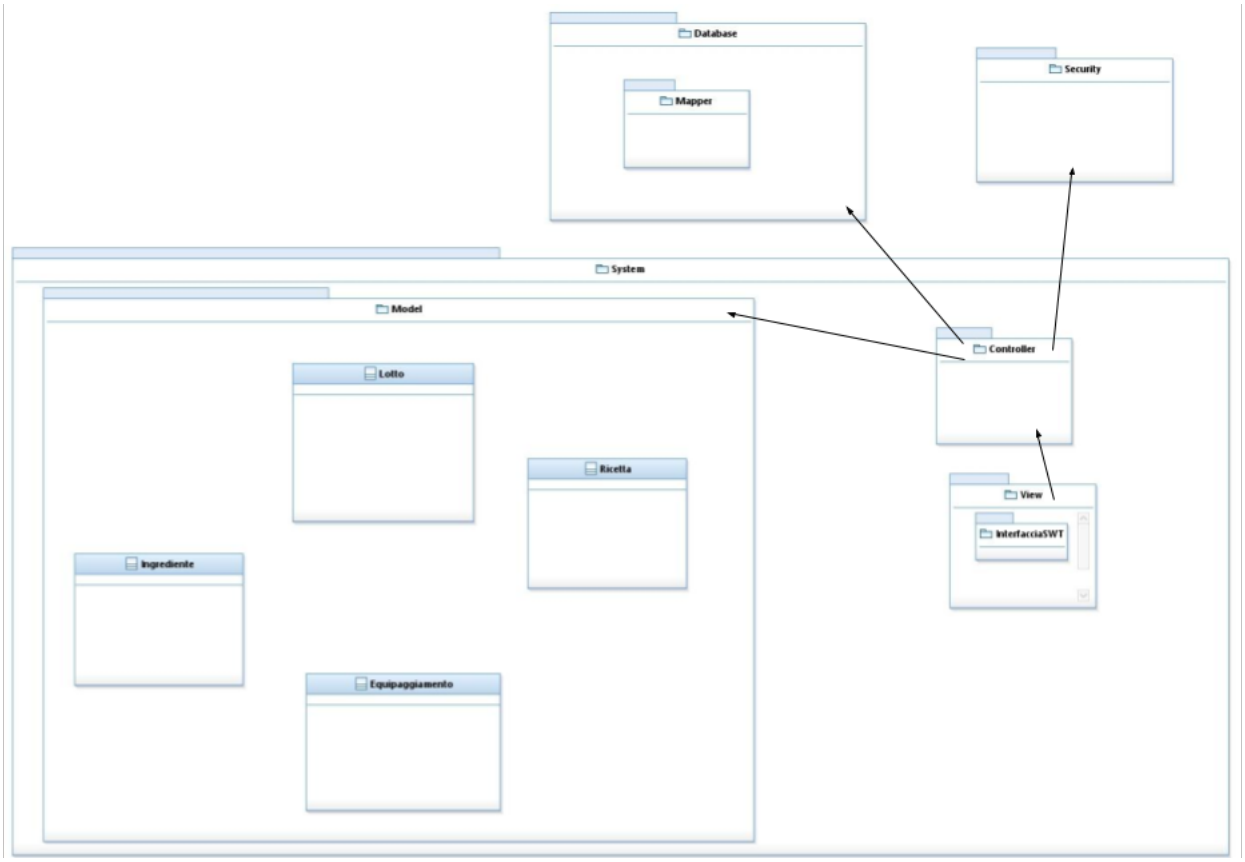
9 Diagrammi di attività

9.1 Diagramma di attività Birra del Giorno



10 Diagramma dell'architettura software

L'architettura di Brew Day è stata progettata secondo un classico MVC a cui è stato aggiunto un layer Database per la memorizzazione dei dati.



11 Diagramma delle classi

Il Diagramma delle classi per ragioni di dimensioni grafiche è stato allegato in esterno nella cartella documentazione.

12 Design principles

12.1 Liskov Substitution Principle

Nell'ambito della funzionalità delle note aggiunte per ogni lotto prodotto non è stata effettuata una divisione netta tra note senza valutazione e note con valutazione, queste ultime infatti sono una sottoclasse delle prime. Secondo il principio LSP per qualsiasi operazione svolta dall'utente su nota senza valutazione essa è valida anche per oggetti di tipo nota con valutazione.

12.2 Acyclic Dependencies Principle

Secondo il principio ADP è buona pratica evitare cicli tra diversi package, Brew Day applica questo principio in maniera pulita, anche grazie all'architectural pattern MVC.

13 Architectural patterns

13.1 MVC

Per la progettazione del sistema Brew Day si è adottato il pattern architetturale MVC che separa la vista con cui l'applicazione comunica con l'utente (View) dalla gestione logica dei comandi (controller) e dai metodi per accedere ai dati (model). Al pattern MVC sono stati inoltre aggiunti due layer: Database e Security.

Il layer Database si occupa di memorizzare e salvare tutte le informazioni riguardanti l'applicazione e i dati utili all'utente (ricette, ingredienti, equipaggiamento). Poichè la quantità di informazioni gestite dal layer database non è eccessivamente grande si è deciso di evitare un approccio con server locale (es. MySql), si è deciso invece di creare un file locale contenente tali informazioni attraverso SQLite. Si tratta di una libreria software che implementa un DBMS, ma a differenza di MySql è molto compatta e veloce, ottima per applicazioni di questo tipo. SQLite permette la creazione di un file locale a cui è possibile accedere solo tramite l'applicazione Brew Day e che svolge il compito di Database.

Il layer security si occupa dell'autenticazione dell'utente.La prima volta che l'utente accede all'applicazione prima di svolgere qualsiasi operazione deve registrare una password personale. Una volta inserita la password il layer security si occupa di creare un file locale criptato (security.bd) contentente la password che l'utente dovrà inserire ogni volta che effettuerà l'accesso.

Il motivo per cui è stato deciso di applicare questo pattern è che è molto raccomandato per questo tipo di applicazioni, inoltre in assenza di numerosi package garantisce uno schema architetturale pulito ed efficiente.

13.2 Data mapper

L'applicazione Brew Day fa largo uso del Data Source Architectural Pattern Data Mapper. All'interno del layer database sono state infatti definite le classi MapperIngrediente, MapperRicetta, MapperLotto e MapperNota. Queste classi servono da intermediari tra gli oggetti nel model e le tabelle del database mantenendo indipendenza tra oggetti in memoria dell'applicazione e oggetti del database.

Il motivo per cui è stato adottato il Data Mapper è che si tratta di un pattern architetturale molto diffuso ed estremamente utile per applicazioni che fanno uso di database.

13.3 Dependent Mapping

In relazione al Data Mapper è stato sfruttato anche il pattern Dependent Mapping che con efficienza. Tramite un'unica classe Mapper, esso gestisce le operazioni CRUD a livello database di una classe figlia che rappresenta una collezione di oggetti della classe parent. In questo caso la classe parent è Ricetta, essa infatti per ogni oggetto contiene una collezione di tipo Quantità. Tutte le operazioni di lettura, aggiornamento e eliminazione degli oggetti quantità vengono effettuate nella classe MapperRicetta.

Il motivo per cui è stato applicato questo pattern è la semplicità con cui vengono gestite le operazioni CRUD della classe figlia, inoltre ha permesso di evitare l'aggiunta di un Mapper apposito che avrebbe appesantito il codice.

14 Design patterns

14.1 Observer

Nell'ambito dell'intefaccia è stato implementato il pattern Observer che notifica le viste delle ricette e degi ingredienti. Ogni volta che viene aggiunto un ingrediente o una ricetta il pattern Observer informa la view che aggiorna la lista degli oggetti disponibili con i nuovi elementi.

Il motivo principale per cui è stato applicato questo pattern è che permette di evitare l'aggiunta di metodi macchinosi che diminuiscono la qualità del sistema (es. aggiunta di un pulsante "Aggiorna").

15 Tecnologie utilizzate

15.1 SQLite

SQLite è una libreria software che implementa un DBMS incorporandolo nell'applicazione stessa. E' un approccio molto più leggero è semplice rispetto all'implementazione di un server con MySql, perfetto per piccole applicazione. La tecnologia di SQLite permette la creazione di un file locale che svolge la funzione di database. Nello specifico di Brew Day tale database è accessibilie solo attraverso una password criptata contenuta in un file security, in modo tale da evitare che chiunque non sia proprietario dell'applicazione possa compiere azioni illegittime.

15.2 install4j

install4j è un software professionale che partendo da un codice sorgente permette di generare un file eseguibile del programma che illustra all'utente finale i passi per l'installazione. Essendo un software a pagamento per la costruzione dell'eseguibile di Brew Day è stata utilizzata la prova gratuita di 90 giorni.

15.3 JUnit

JUnit è un framework di unit testing per il linguaggio di programmazione Java con il quale sono stati costruiti i test di Brew Day per raggiungere una coverage affidabile.

15.4 SWT

SWT è una libreria Java per la programmazione di interfacce grafiche. Attraverso la modellazione manuale del lay out grafico SWT genera il rispettivo codice Java.

16 Sicurezza

Visto che la nostra applicazione è esclusivamente locale e non si appoggia a server esterni abbiamo adottato misure volte a proteggere i dati inseriti.

1. Database criptato con password;
2. Salvataggio dell' hash della password per regolare l'accesso.

Lo svantaggio di questa soluzione è che non è possibile recuperare i dati in caso di perdita della password.

17 Github

17.1 Branches

Nel corso dello sviluppo di Brew Day il gruppo ha lavorato molto sul branch master in quanto non vi è stata una divisione netta del lavoro ma piuttosto un'approccio collaborativo in parallelo. Sono stati però creati tre branch per lo sviluppo di alcune feature:

- feature-bod: branch nel quale è stato sviluppata interamente la funzione Brew of the Day
- feature-is: branch nel quale è stato sviluppata interamente la funzione "Lista della spesa"
- refactor-swt: branch nel quale è stato ripulito il codice generato in automatico da SWT (paragrafo 15).

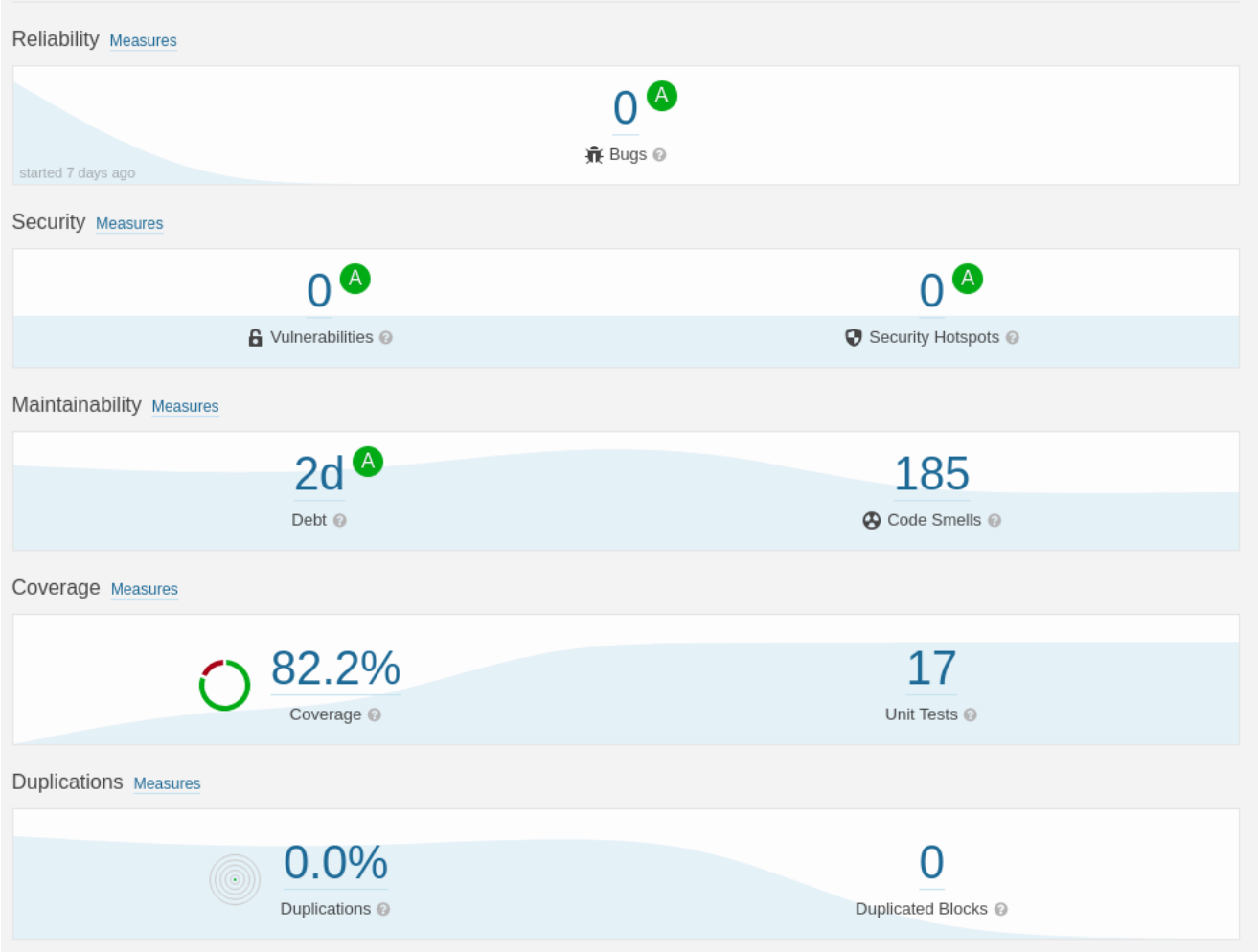
17.2 tags

I git tags create durante lo sviluppo di Brew Day sono i seguenti:

- v0.1 -ing: Primo tag in assoluto in cui si sono impostate le basi dell'applicazione (Database, intrfaccia grafica, ...) e in cui è stata completata la prima feature, ovvero la gestione degli ingrediente (Aggiunta, rimozione, aggiornamento).
- v0.2: secondo tag in cui è stata implementata la funzionalità principale di Brew Day, ossia la gestione delle ricette.
- v0.3 -lotto Terzo tag in cui sono stati inserite ulteriori feature tra cui la traccia dei lotti di birra prodotti, la lista della spesa che notifica la scarsità o meno degli ingredienti e la funzione Brew of The Day che restituisce la ricetta che massimizza l'uso degli ingredienti.
- v1.0 -consegna: Ultimo tag definitivo in cui sono state implementate ulteriori funzioni, tra cui l'aggiornamento automatico degli ingredenti e in cui sono stati corretti problemi riguardo a bugs e codice duplicato e abbellita la grafica dell'applicazione.

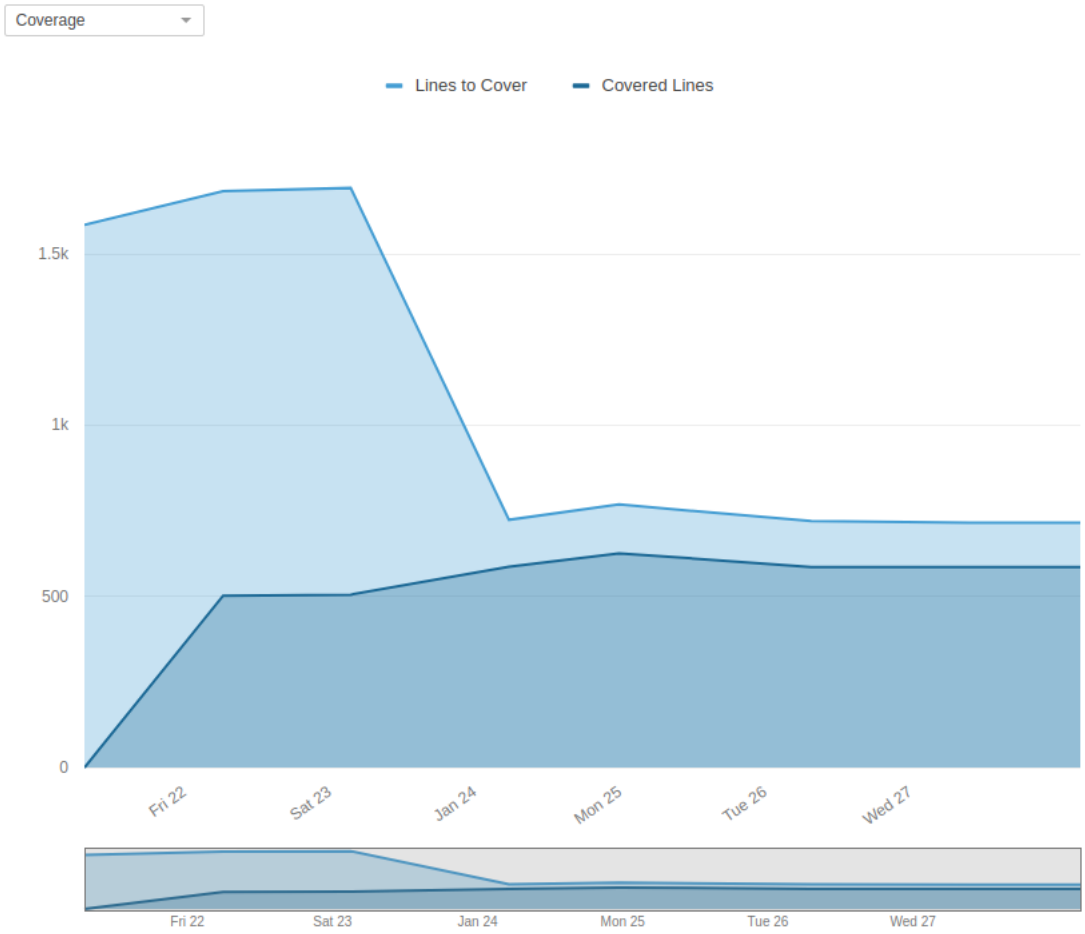
18 Sonarcloud

Inizialmente abbiamo effettuato le analisi localmente utilizzando il software Sonarqube, successivamente ci siamo spostati sulla versione cloud. Le analisi ed eventuali correzioni sono state effettuate alla fine di ogni processo di sviluppo delle principali features.
I voti ottenuti alla fine del progetto sono i seguenti:



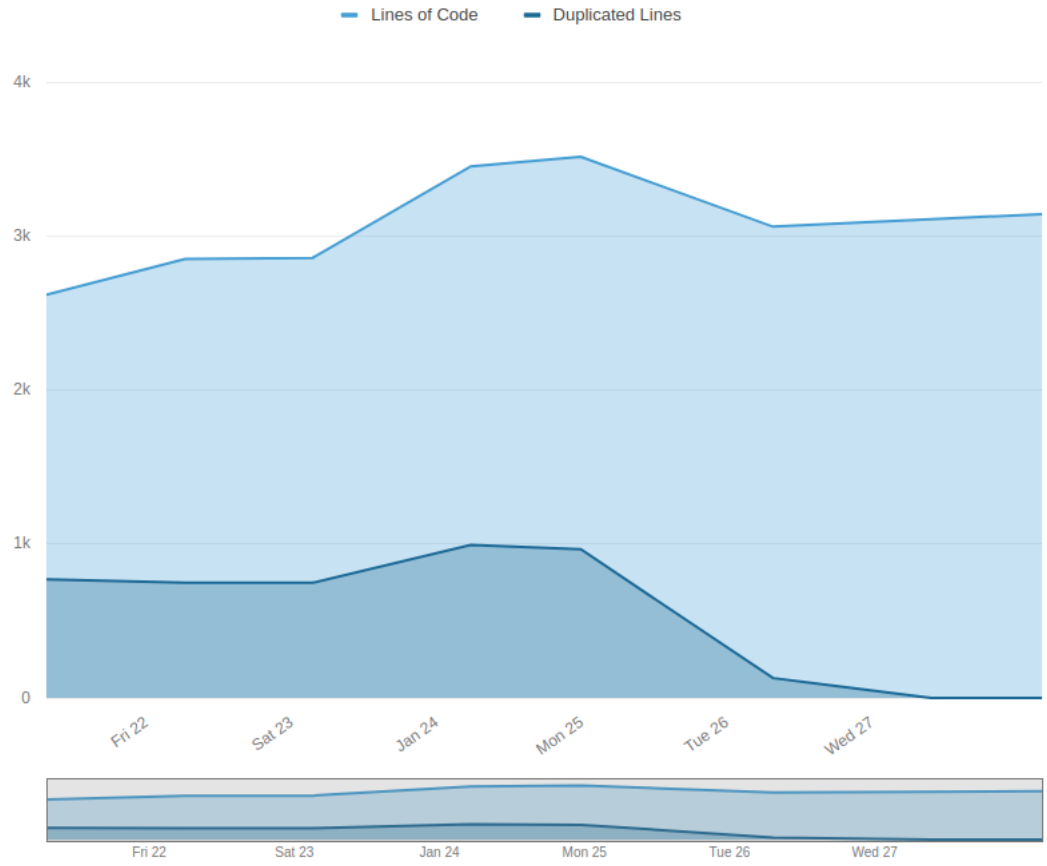
18.1 Coverage

Il risultato finale della coverage si aggira intorno all' 82%.
Tuttavia abbiamo deciso di escludere dall'analisi di coverage il package view, in quanto esso è principalmente codice per la generazione dell'interfaccia grafica e praticamente nessuna logica applicativa.
Dal grafico temporale riportato è possibile vedere la forte riduzione di linee di codice da testare.



18.2 Duplication

Inizialmente il codice del nostro progetto soffriva di un alta duplicazione (superiore al 22%) dovuta principalmente all codice generato in automatico per le classi della GUI in SWT. È stato succesivamente effettuato un refactor del codice per ridurre la duplicazione.
Il risultato finale è una duplicazione dello 0%.

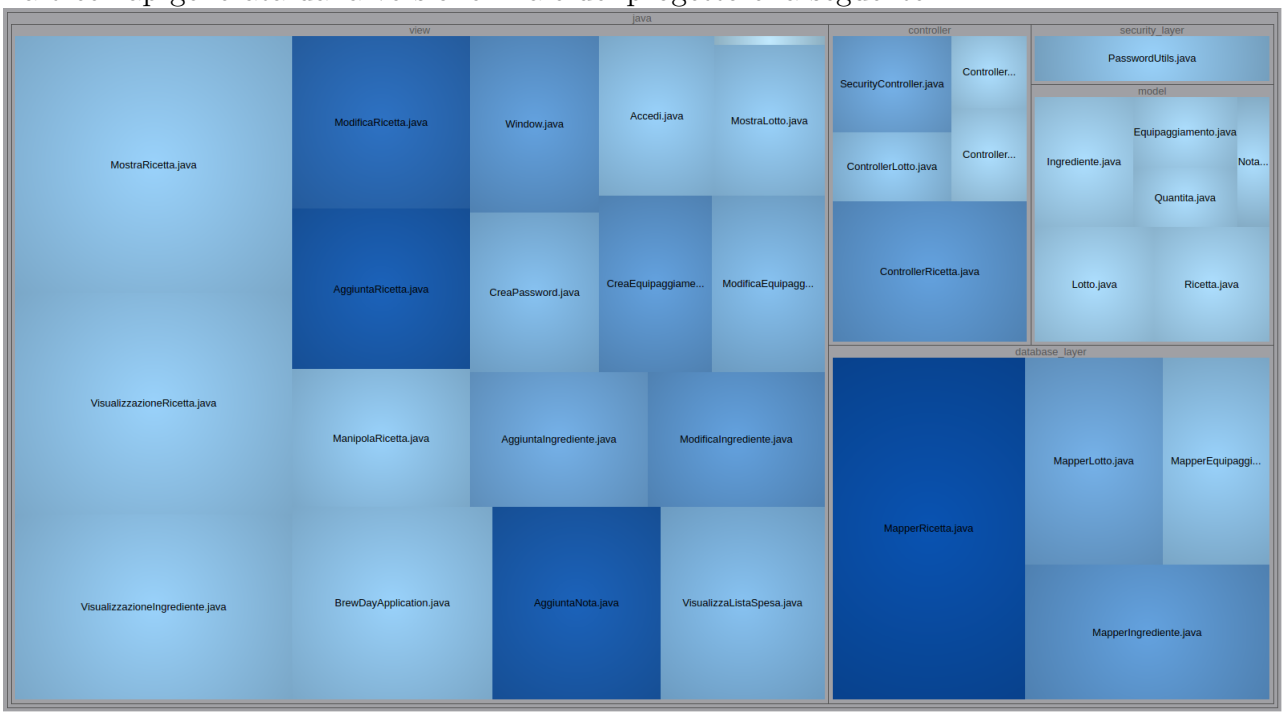


19 Understand

Durante lo svolgimento del progetto è stato utilizzato il tool Understand per analizzare la struttura del nostro software.

19.1 Treemap

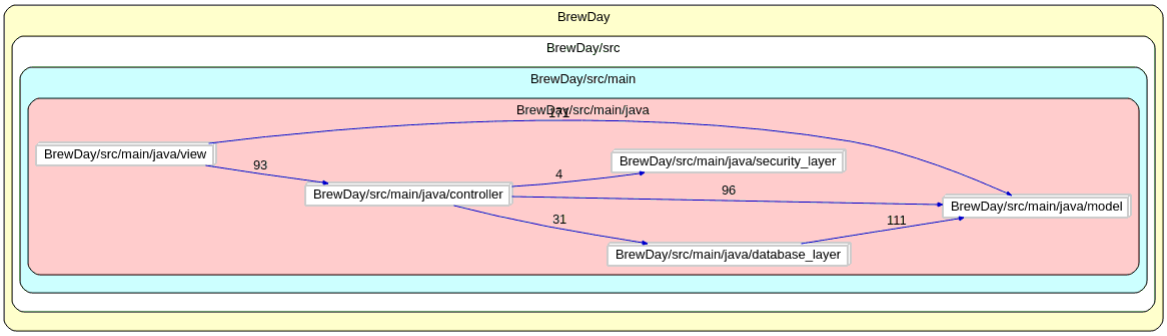
La treemap generata dalla versione finale del progetto è la seguente:



Si può notare come le classi con complessità meggiore sono quelle riguardanti le classi che hanno responsabilità sulle funzioni principali del software: il MapperRicetta e le classi della GUI riguardanti le ricette e le note dei lotti.
Viste le dimensioni ridotte del progetto non abbiamo ritenuto neccessario intervenire per ridurre la complessità.

19.2 Grafo delle dipendenze

. Il grafo delle dipendenze riguardante la nostra architettura software:



Viste le dimensioni non abbiamo rilevato la presenza di Antippatern Architteturali. Tuttavia il package controller ha dipendenze su ben altri 3 package (model,database_layer e security_layer). Per ora non lo abbiamo considerato un grande problema, ma con l'aumentare delle dimensioni del software potrebbe diventare un External Breakable.