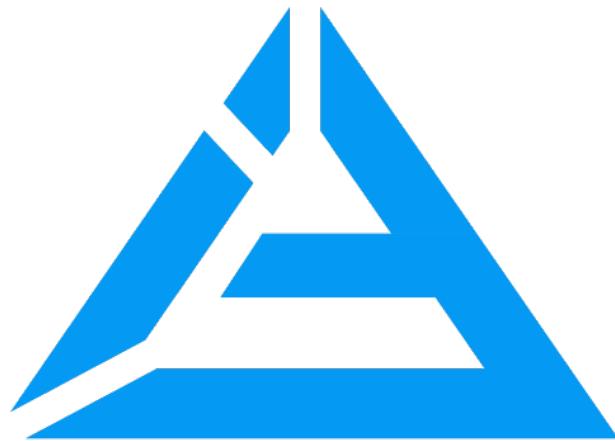




Laurea Magistrale in informatica-Università di Salerno  
Corso di *Ingegneria, Gestione ed Evoluzione del Software* - Prof. A. De  
Lucia



**FITDIARY**  
DECIDE, COMMIT, SUCCEED

## MASTER\_TP

### Master Test Plan

<b>Versione</b>	1.0
<b>Data</b>	17/10/2023
<b>Destinatario</b>	Prof.re Andrea De Lucia
<b>Presentato da</b>	Aurelio Sepe, Simone Spera, Otino Pio Santosuosso, Raffaele Sais
<b>Approvato da</b>	



## Sommario

1. Obiettivi .....	3
1.1 Contesto.....	3
2. Strategie.....	3
2.1 Testing di Unità .....	3
2.2 Testing di Integrazione.....	3
2.3 Testing di Sistema .....	3
2.4 Testing di Regressione .....	4
3. Attività di Test.....	4
3.1 Attività .....	4
3.2 Classi Testate .....	4
3.2.1 Testing di Unità .....	4
3.2.2 Testing di Integrazione.....	4
3.3 Classi Non Testate .....	5
4. Test di Regressione .....	5



## 1. Obiettivi

---

### 1.1 Contesto

FitDiary fornisce supporto ai preparatori e ai clienti che devono gestire il proprio piano alimentare e di allenamento.

L'obiettivo del testing è quello di identificare quanti più problemi possibili per ottenere e garantire qualità agli utilizzatori del servizio.

## 2. Strategie

---

### 2.1 Testing di Unità

Nel contesto del testing di unità, opteremo per un approccio "white-box", poiché i test sono stati progettati sulla base del codice sorgente anziché sulla specifica, con l'obiettivo di massimizzare la branch coverage. Per garantire un'esecuzione isolata della componente soggetta a test, faremo uso di Mockito per creare i mock necessari.

### 2.2 Testing di Integrazione

Per il testing di integrazione, esamineremo l'interazione tra due o più componenti e la modalità di trasferimento delle informazioni o dei dati tra di esse. In particolare, seguendo un approccio Bottom-up, procederemo con i test sulle classi di tipo "Service" inizialmente. Successivamente, testeremo le classi di tipo "Controller" che dipendono dalle classi di tipo "Service", verificando attentamente che lo scambio di dati tra le due categorie non conduca a errori inaspettati.

### 2.3 Testing di Sistema

Per il testing di sistema è stato analizzato il comportamento complessivo di quest'ultimo, valutando che soddisfatti i requisiti specificati inizialmente. Per determinare i casi di test abbiamo utilizzato la tecnica del Category Partition



## 2.4 Testing di Regressione

La decisione è stata quella di eseguire nuovamente tutti i casi di test riguardanti le componenti impattate dalle Change Request. Questo è stato fatto allo scopo di individuare eventuali anomalie aggiuntive e di assicurarsi che fosse raggiunta una percentuale accettabile di branch coverage.

## 3. Attività di Test

---

### 3.1 Attività

Per garantire una corretta esecuzione del test, saranno eseguite, in ordine, le seguenti attività:

- Realizzazione dei Casi di Test
- Esecuzione dei Test
- Report dei Risultati
- Analisi e Gestione dei Difetti

### 3.2 Classi Testate

Di seguito sono elencate le classi testate e la tipologia di testing:

#### 3.2.1 Testing di Unità

- *GestioneAlimentoServiceImpl.java*
- *GestioneAlimentoConsumatoServiceImpl.java*
- *GestioneCategoriaEsercizioServiceImpl.java*
- *GestioneIstanzaEsercizioEseguitoServiceImpl.java*
- *GestioneEsercizioService.java*
- *GestioneProtocolloServiceImpl.java*
- *GestioneSchedaAlimentareImpl.java*
- *GestioneSchedaAllenamentoService.java*

#### 3.2.2 Testing di Integrazione

- *GestioneAlimentoControllerIntegration.java*
- *GestioneAlimentoConsumatoControllerIntegration.java*
- *GestioneCategoriaEsercizioControllerIntegration.java*
- *GestioneIstanzaEsercizioEseguitoControllerIntegration.java*
- *GestioneProtocolloControllerIntegrazione.java*
- *GestioneSchedaAlimentareControllerIntegration.java*
- *GestioneSchedaAllenamentoControllerIntegration.java*




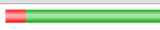
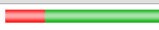
### 3.3 Classi Non Testate

La classe *GestioneProtocolloServiceImpl.java* non è stata testata in quanto non sono state apportate modifiche.


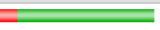
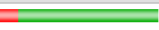
## 4. Test di Regression

Conformemente alle strategie dei test precedentemente delineate, abbiamo seguito l'approccio di ritestare tutti i test di unità e integrazione delle componenti impattate dalle change request, in questo caso la componente impattata che precedentemente aveva dei test è gestione protocollo. I risultati complessivi, come si può vedere dalle immagini sottostanti, sono altamente promettenti e presentano un notevole valore aggiunto per la qualità del software in oggetto.

#### it.fitdiary.backend.gestioneprotocollo.service

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 <a href="#">GestioneProtocolloServiceImpl</a>		87%		75%	4	18	4	48	0	10	0	1
Total	24 of 188	87%	4 of 16	75%	4	18	4	48	0	10	0	1

#### it.fitdiary.backend.gestioneprotocollo.controller

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 <a href="#">GestioneProtocolloController</a>		84%		85%	5	24	23	103	1	10	0	1
Total	57 of 357	84%	4 of 28	85%	5	24	23	103	1	10	0	1

Come evidenziato chiaramente nel rapporto finale, la copertura di Branch ha raggiunto un notevole 80%, superando il limite prefissato del 75%.

Un'analisi più dettagliata dei singoli pacchetti e delle relative statistiche rivela un risultato altamente soddisfacente come è possibile osservare dall'immagine sottostante, dove è riportata la coverage di tutte le componenti del sistema.



Laurea Magistrale in informatica-Università di Salerno  
Corso di *Ingegneria, Gestione ed Evoluzione del Software* - Prof. A. De Lucia

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
it.fitdiary.backend.gestioneReport.controller		65%		58%	8	18	34	85	1	6	0	1
it.fitdiary.backend.entity.enums		0%		n/a	2	2	16	16	2	2	2	2
it.fitdiary.backend.gestioneStimaProgressi.adapter		8%		0%	3	4	16	20	1	2	0	1
it.fitdiary.backend.gestioneAlimentoConsumato.controller		63%		35%	9	15	21	56	0	5	0	1
it.fitdiary.backend.gestioneAbbonamento.controller		9%		0%	3	5	24	26	1	3	0	1
it.fitdiary.backend.gestioneAlimentoConsumato.service		60%		40%	5	10	15	37	1	5	0	1
it.fitdiary.backend.gestioneutenza.controller		84%		87%	3	18	17	108	2	14	0	1
it.fitdiary.backend.gestioneProtocollo.controller		84%		85%	5	24	23	103	1	10	0	1
it.fitdiary.backend.gestioneSchedaAlimentare.controller		76%		75%	6	15	19	64	2	7	0	1
it.fitdiary.backend.gestioneSchedaAlimentare.service		86%		66%	6	16	8	61	1	7	0	1
it.fitdiary.backend.gestioneSchedaAllenamento.controller		82%		75%	5	15	16	69	1	7	0	1
it.fitdiary.backend.gestioneSchedaAllenamento.service		86%		64%	6	14	6	57	1	7	0	1
it.fitdiary.backend.utility.service		89%		62%	5	28	8	85	2	24	0	3
it.fitdiary.backend.gestioneutenza.service		93%		93%	5	44	7	98	1	11	0	1
it.fitdiary.backend.gestioneEsecuzioneEsercizio.controller		75%		60%	4	9	11	36	0	4	0	1
it.fitdiary.backend.gestioneProtocollo.service		87%		75%	4	18	4	48	0	10	0	1
it.fitdiary.backend.gestioneEsecuzioneEsercizio.controller.dto		74%		n/a	1	19	8	36	1	19	0	2
it.fitdiary.backend.gestioneAlimentoConsumato.controller.dto		83%		n/a	1	16	5	32	1	16	0	2
it.fitdiary.backend.gestioneEsecuzioneEsercizio.service		93%		50%	3	7	2	32	0	4	0	1
it.fitdiary.backend.gestioneAlimento.service		73%		50%	2	6	2	9	0	4	0	1
it.fitdiary.backend.utility		95%		86%	3	15	2	40	0	4	0	2
it.fitdiary.backend.OnStartup		95%		75%	1	4	2	33	0	2	0	1
it.fitdiary.backend.gestioneReport.service		93%		100%	1	10	2	22	1	6	0	1
it.fitdiary.backend.gestioneStimaProgressi.service		70%		n/a	1	3	1	4	1	3	0	1
it.fitdiary.backend		58%		n/a	1	3	2	4	1	3	0	1
it.fitdiary.backend.gestioneSchedaAllenamento.controller.dto		100%		n/a	0	32	0	65	0	32	0	3
it.fitdiary.backend.gestioneSchedaAlimentare.controller.dto		100%		n/a	0	24	0	48	0	24	0	3
it.fitdiary.backend.gestioneAlimento.controller		100%		n/a	0	4	0	14	0	4	0	1
it.fitdiary.backend.gestioneCategorieEsercizio.controller		100%		100%	0	5	0	10	0	4	0	1
it.fitdiary.backend.gestioneEsercizio.controller		100%		100%	0	5	0	10	0	4	0	1
it.fitdiary.backend.gestioneCategorieEsercizio.service		100%		n/a	0	4	0	4	0	4	0	1
it.fitdiary.backend.gestioneEsercizio.service		100%		n/a	0	4	0	4	0	4	0	1
Total	979 of 4.974	80%	81 of 304	73%	93	416	271	1.336	22	261	2	42

Quasi la totalità dei Branch delle componenti coinvolte, gestite dalle suite di test a disposizione del team, è stata adeguatamente coperta. Questo dimostra l'efficacia e la completezza delle attività di testing svolte durante la fase di manutenzione del progetto.

In sintesi, i risultati confermano il successo dell'approccio di testing e la validità della strategia adottata per garantire la qualità del software, mantenendo una copertura di Branch al di sopra delle aspettative, nonostante alcune specifiche scelte progettuali.