

**Università degli Studi di Salerno**  
Corso di Ingegneria del Software

**AutoFlow**  
**Test Regression Report**  
**Versione 1.0**



Data: 08/11/2025

## **Sommario**

1. Introduzione .....	3
2. Relazione con gli altri documenti .....	3
3. Test Log .....	4
3.1. TC5 .....	4
4. Glossario .....	10

## **1. Introduzione**

Questo documento presenta i risultati di una sessione di test condotta sul sistema AutoFlow, con l'obiettivo di verificare che il comportamento dell'applicazione sia conforme alle funzionalità richieste nelle fasi di analisi e progettazione. La suite eseguita deriva dai casi di test descritti nella TCS (Test Case Specification) ed è stata applicata alla versione attuale del backend e alle principali funzionalità del flusso utente. Il Test Execution Report raccoglie gli esiti dei casi eseguiti, le condizioni di test, gli eventuali problemi rilevati e una sintesi della copertura del codice ottenuta. L'intento è fornire una visione chiara e strutturata dello stato di qualità del software, utile sia per il team di sviluppo che per la pianificazione delle fasi successive.

## **2. Relazione con gli altri documenti**

Il presente progetto prende forma a partire dall'analisi e dal confronto con soluzioni digitali già affermate nel settore della gestione dei processi commerciali e configurativi in ambito automotive, che rappresentano un riferimento consolidato in termini di efficienza e affidabilità.

Di seguito si presenta un elenco dei documenti chiave del progetto a cui si fa esplicito riferimento:

- Test Plan (TP): documento di pianificazione della fase di test del sistema.
- Test Case Specification (TCS): documento che descrive in dettaglio una serie di test case, specificando input, azioni, condizioni di esecuzione e risultati attesi.
- Test Execution Report: documento di log dell'esecuzione dei test.

Oltre ai documenti del progetto, si fa riferimento ad opere di letteratura tecnica che hanno contribuito allo sviluppo metodologico e concettuale di questo lavoro:

- Object-Oriented Software Engineering Using UML, Patterns, and Java™ Third Edition di Bernd Bruegge & Allen H. Dutoit.

### 3. Test Log

In questa sezione è mostrato il risultato del test di regressione eseguito su TC5. L'intera suite è stata nuovamente testata ma per brevità sono omessi i risultati degli altri TC, essendo tutti **Passati**.

#### 3.1. TC5

A seguito del fallimento dei test relativi al caso TC5 è stata avviata una fase di debugging per individuare l'origine dell'errore. L'analisi del test unitario ha evidenziato l'utilizzo di stubbing superflui in *FatturaServiceImpl*, dovuti a una configurazione dei mock non perfettamente allineata alla logica effettiva del metodo testato. Ciò ha impedito al test di riflettere correttamente il comportamento reale del servizio.

Parallelamente, il test di integrazione nel *FatturaControllerIT* ha prodotto una *DataIntegrityViolationException*, indicando che alcuni campi obbligatori dell'entità non venivano valorizzati durante il salvataggio del documento PDF. Questa incongruenza non era stata rilevata nei test unitari, poiché in quel contesto la persistenza e la validazione JPA non erano attive.

Una volta identificate le cause, la revisione dei mock e della logica di salvataggio ha consentito di correggere il malfunzionamento e completare l'esecuzione del caso TC5 senza ulteriori anomalie.

#### Metodi corretti

##### Sezione di codice tc5\_01

```
124 // mock PDF generato
125 DocumentoPDF doc = new DocumentoPDF();
126 doc.setNameFile("fattura-777.pdf");
127 doc.setPath("/tmp/fattura-777.pdf");
128 doc.setDimensioneBytes(12345L);
129 doc.setaDataCreazione(LocalDateTime.now());
130 doc.setUltimaModifica(LocalDateTime.now());
131 doc = documentoPDFRepository.save(doc);
132
133 given(pdfDocumentService.generateInvoicePdf(anyLong()))
134 .willReturn(doc);
135
136 mockMvc.perform(post(urlTemplate: "/api/fatture/da-proposta/{propostaId}", p.getId())
137 .contentType(MediaType.MediaType.APPLICATION_JSON))
138 .andExpect(status().isCreated())
139 .andExpect(content().contentTypeCompatibleWith(MediaType.MediaType.APPLICATION_JSON))
140 .andExpect(jsonPath(expression: "$.id", greaterThan(value: 0)))
141 .andExpect(jsonPath(expression: "$.clienteId").value(cliente.getId()))
142 .andExpect(jsonPath(expression: "$.propostaId").value(p.getId()))
143 .andExpect(jsonPath(expression: "$.importoTotale").value(expectedValue: 25_000.0));
144
145 List<Fattura> all = fatturaRepository.findAll();
146 assertEquals(expected: 1, all.size());
147 Fattura f = all.get(0);
148 assertEquals(cliente.getId(), f.getCliente().getId());
149 assertEquals(p.getId(), f.getProposta().getId());
150 }
```

### Sezione di codice tc5\_04

```
191     if (f.getId() == null) {
192         f.setId(200L);
193     }
194     return f;
195 });
196
197 //given(fatturaRepository.findById(200L)).willReturn(Optional.of(new Fattura()));
198
199 // simuliamo errore del File System tramite PdfDocumentService
200 given(pdfDocumentService.generateInvoicePdf( fatturald: 200L))
201     .willThrow(new IllegalStateException("Impossibile generare il PDF - spazio i
202
203 IllegalStateException ex = assertThrows(
204     IllegalStateException.class,
```

## Unit test

I test unitari hanno riguardato il *FatturaService* e hanno verificato che la fattura venga creata solo quando la proposta risulta confermata, che il totale calcolato sia corretto e che il PDF venga generato nel percorso previsto.

Tutti i test di unità sono **Passati** (*Tranne il tc5\_04 che è stato risolto in seguito*).

The screenshot shows two windows of the IntelliJ IDEA Test Runner.

The top window displays a failed test: **FatturaServiceImplTest (it.autoflow.invoice.service) tc5\_04\_spazioInsufficienteDuranteGenerazionePdf**. The test took 459 ms and failed. The error message is:

```
1 test failed 1 test total, 459 ms
/Users/zgiov/Library/Java/JavaVirtualMachines/corretto-17.0.17/Contents/Home/bin/j
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classe
org.mockito.exceptions.misusing.UnnecessaryStubbingException:
Unnecessary stubbings detected.
Clean & maintainable test code requires zero unnecessary code.
Following stubbings are unnecessary (click to navigate to relevant line of code):
  1. -> at it.autoflow.invoice.service.FatturaServiceImplTest.tc5_04_spazioInsuffi
Please remove unnecessary stubbings or use 'lenient' strictness. More info: javado
at org.mockito.junit.jupiter.MockitoExtension.afterEach(MockitoExtension.java:...
```

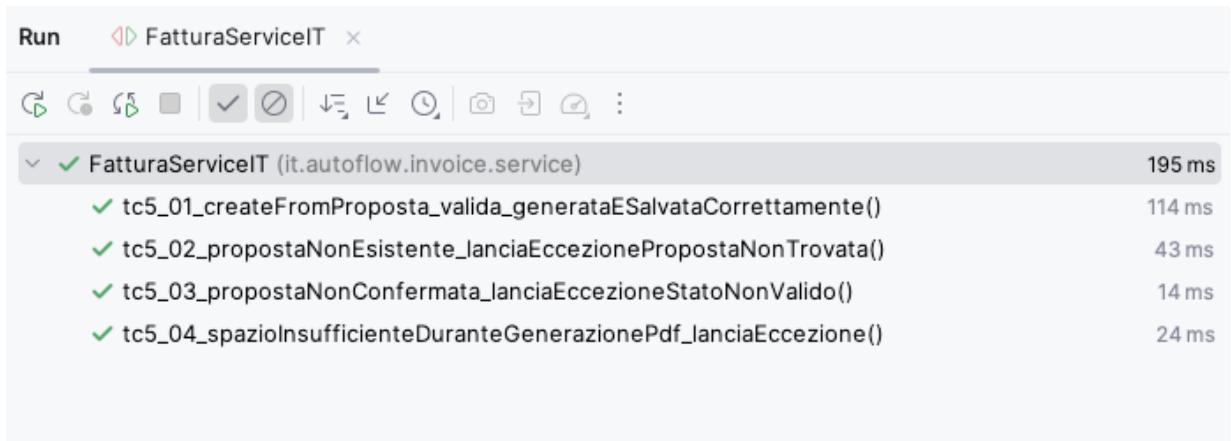
The bottom window shows a successful run of the same test class. The test **FatturaServiceImplTest** passed all its tests:

Test	Time
tc5_01_createFromProposta_valida_generataESalvataCorrettamente()	496 ms
tc5_02_propostaNonEsistente_lanciaEccezionePropostaNonTrovata()	2 ms
tc5_03_propostaNonConfermata_lanciaEccezioneStatoNonValido()	1 ms
tc5_04_spazioInsufficienteDuranteGenerazionePdf_lanciaEccezione()	3 ms

## Integration service con repository (H2)

I test hanno confermato il corretto salvataggio della fattura con la relativa associazione alla proposta, il sollevamento dell'errore previsto nel caso in cui la proposta non risulti confermata e la corretta gestione delle eccezioni di I/O in presenza di problemi di scrittura.

Tutti i test risultano Passati.



## Integration controller

MockMvc ha verificato la restituzione dello stato HTTP 201 insieme al path logico del PDF prodotto e il comportamento corretto del sistema negli scenari di errore previsti.

Tutti i test sono **Passati** (*Tranne il tc5\_01 che è stato risolto in seguito*).

The screenshot shows two windows of the IntelliJ IDEA Test Runner for the class `FatturaControllerIT`.

**Top Window (Failed Test):**

- Run tab selected.
- Test: `FatturaControllerIT (it.autoflow.invoice.controller)` took 105 ms.
- Test Result: 1 test failed, 1 passed, 2 tests total, 105 ms.
- Failed Test: `tc5_01_createFromPropostaValida_restuisce201EJson` took 59 ms.
- Passed Test: `tc5_02_createFromPropostaInesistente_lanciaEntityN` took 46 ms.
- Stack Trace:

```
2025-12-05T11:06:15.448+01:00  WARN 23366 --- [           main] o.h.engine.jdbc.spi.SqlExceptionHelper : insert into documenti_pdf (data_creazione,dimensione_bytes,nome_file,path,ultima_modifica) values (?, ?, ?, ?, ?)
```

```
org.springframework.dao.DataIntegrityViolationException: could not execute statement
```

```
insert into documenti_pdf (data_creazione,dimensione_bytes,nome_file,path,ultima_modifica)
```

```
at org.springframework.orm.jpa.vendor.HibernateJpaDialect.convertHibernateAccessException(HibernateJpaDialect.java:233)
```

```
at org.springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIfPossible(HibernateJpaDialect.java:214)
```

```
at org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.translateExceptionIfPossible(AbstractEntityManagerFactoryBean.java:401)
```

```
at org.springframework.dao.support.ChainedPersistenceExceptionTranslator.translateExceptionIfPossible(ChainedPersistenceExceptionTranslator.java:59)
```

**Bottom Window (All Passed):**

- Run tab selected.
- Test: `FatturaControllerIT (it.autoflow.invoice.controller)` took 179 ms.
- Test Results:
  - `tc5_01_createFromPropostaValida_restuisce201EJson()` took 166 ms.
  - `tc5_02_createFromPropostaInesistente_lanciaEntityNotFoundException()` took 13 ms.

## **Test di sistema**

*Per i casi di test TC4 e TC5 sono stati eseguiti integralmente i test di unità e di integrazione; tuttavia, non è stato possibile automatizzare tramite Selenium alcuni scenari specifici, poiché l'interfaccia del sistema impedisce intenzionalmente la creazione di input o condizioni non previste. Tali limitazioni lato front-end impediscono la riproduzione artificiale di alcune situazioni di errore, già prevenute dal design applicativo.*

#### 4. Glossario

Termine	Definizione
Test unitario	Verifica del comportamento di singole unità di codice, generalmente metodi di servizio, mediante l'uso di mock per isolare le dipendenze.
Test di integrazione	Controllo del comportamento complessivo di più componenti del backend, comprendendo servizi, repository, transazioni e controller, spesso tramite database H2 in-memory.
Test di sistema	Validazione del comportamento percepito dall'utente finale attraverso l'interazione con il frontend, simulata grazie a Selenium IDE.
Oracolo di test	Risultato atteso definito nella specifica dei casi di test, utilizzato come riferimento per valutare l'esito della prova.
Continuous Integration	Processo automatizzato che esegue build, test e verifiche di qualità a ogni modifica del codice, garantendo stabilità e riduzione delle regressioni.
Code Coverage	Misura del grado di esercizio del codice sorgente da parte della suite di test, considerando classi, metodi, linee e branch.
Mock	Oggetto simulato utilizzato per isolare una dipendenza e verificare la logica di una singola unità in condizioni controllate.
H2 in-memory	Database leggero e transitorio utilizzato durante i test di integrazione, che consente esecuzioni rapide senza necessità di un'istanza esterna.
MockMvc	Strumento di Spring utilizzato per testare gli endpoint REST simulando richieste HTTP senza avviare un server reale.
Selenium IDE	Strumento usato per la creazione e l'esecuzione di script automatizzati che riproducono i comportamenti dell'utente nel browser.