



Università degli Studi di Salerno

Corso di Ingegneria del Software
Classe 1 Resto 0
Corso di Laurea in Informatica
A.A. 2022/23

Quiad Test Execution Report

Versione 2.1
27/12/2022



Partecipanti al progetto e scriventi

Nome	Matricola
Di Pasquale Valerio	0512110638
Troisi Vito	0512109807

Revision History

Data	Versione	Descrizione	Autore
23/12/2022	1.0	Prima stesura Test Execution Report e Indice	T.V.
27/12/2022	2.0	Testing di unità lato server	D.P.V. T.V.
27/12/2022	2.1	Testing REST API: prima parte	T.V.

Indice

1. Overview.....	p.4
2. Riferimenti.....	p.4
3. Report del testing di unità server-side.....	p.5
4. Report del testing della REST API.....	p.17
5. Report del testing di unità client-side.....	p.
6. Report del testing di integrazione.....	p.
7. Report del testing di sistema.....	p.

Overview

Il presente documento riporta i risultati dell'esecuzione dei test effettuati, ed è comprensivo di ogni attività svolta, dal testing di unità a quello di sistema.

Si procederà innanzitutto analizzando i risultati del testing della REST API il che, con riferimento allo Object Design Document, è effettuabile inviando richieste opportune agli endpoint HTTP. Saranno poi osservati i risultati dell'esecuzione dei test di unità, integrazione e sistema. Si precisa che NON saranno riportati gli output di metodi elementari (i.e. getters/setters).

Riferimenti

Per procedere al meglio nella lettura del presente documento, si rimanda a documenti ulteriori legati ai requisiti del sistema Quiad, alla sua progettazione, ed al piano di testing redatto a monte di tale attività. Nello specifico, si segnalano:

- Il Requirements Analysis Document (RAD)
- Il System Design Document (SDD)
- Lo Object Design Document (ODD)
- Il Test Plan

Il Test Plan, il quale contiene, per ogni funzionalità da testare, i casi di test esplicitati mediante tecniche quali *Category Partition*, è di particolare rilievo nel contesto del Test Execution Report presente, ed esplicita inoltre le modalità e gli strumenti impiegati nell'esecuzione stessa dei test.

Report del testing di unità server-side**Unità: Sottosistema Account Management**

AccountController	
Test:	AccountController::findByUsername("quiad")
Esiste:	Account con username = "quiad"
Oracolo:	Restituito account esistente con username "quiad".
Test:	AccountController::findByUsername("quiadz")
Esiste:	Account con username = "quiad"
Oracolo:	Nessun account restituito.
Test:	AccountController::createAccount() <ul style="list-style-type: none">- Username: "quiad"- Email: "quiad@test.com"- Password: "quiad"
Esiste:	-
Oracolo:	Restituito l'account creato.

AuthService

Test:	AuthService::login() - Username: "quiad" - Password: "miapassword" In una mock HTTP request.
Esiste:	Account con username = "quiad"
Oracolo:	La risposta HTTP contiene l'utente il cui username è quiad.
Test:	AuthService::login() - Username: "quiad" - Password: "miapasswordz" In una mock HTTP request.
Esiste:	Account con username = "quiad"
Oracolo:	La risposta HTTP contiene l'errore 401 - Unauthorized.
Test:	AuthService::login() - Username: "daqh" - Password: "miapassword" In una mock HTTP request.
Esiste:	Account con username = "quiad"
Oracolo:	La risposta HTTP contiene l'errore 401 - Unauthorized.
Test:	AuthService::login() - Username: "" - Password: "miapassword" In una mock HTTP request.
Esiste:	Account con username = "daqh"
Oracolo:	La risposta HTTP contiene l'errore 400 - Bad Request.
Test:	AuthService::login() - Username: "daqh" - Password: "" In una mock HTTP request.

Esiste:	Account con username = "daqh"
Oracolo:	La risposta HTTP contiene l'errore 400 - Bad Request.
Test:	AuthService::login() - Username: "daqh" - Password: "" In una mock HTTP request.
Esiste:	Il mock object PrismaClient è istanziato al fine di restituire un errore, simulando un DB non raggiungibile.
Oracolo:	La risposta HTTP contiene l'errore 500 - Server Error.

RegistrationService

Test:	RegistrationService::register() <ul style="list-style-type: none">- Email: "valeriotroisi@quiad.com"- Username: "quiad"- Password: "miapassword" In una mock HTTP request.
Esiste:	-
Oracolo:	La risposta HTTP contiene l'utente registrato.

Test:	RegistrationService::register() <ul style="list-style-type: none">- Email: "valeriotroisi@quiad.com"- Username: "quiad"- Password: "miapassword" In una mock HTTP request.
Esiste:	Account con username = "quiad"
Oracolo:	La risposta HTTP contiene l'errore 500 - Server Error.

Unità: Sottosistema Document Management**DocumentController**

DocumentController	
Test:	DocumentController::findDocument() - OriginPlace: "Salerno"
Esiste:	Documento con: - Name: "Registro delle nascite" - RetrievalDate: 2022-01-01 - RetrievalPlace: "Salerno" - OriginDate: 2022-01-01 - OriginPlace: "Salerno" - Path: "/"
Oracolo:	Restituito il documento esistente.
Test:	DocumentController::createDocument() - Name: "Registro delle nascite" - RetrievalDate: 2022-01-01 - RetrievalPlace: "Salerno" - OriginDate: 2022-01-01 - OriginPlace: "Salerno" - Path: "/"
Esiste:	-
Oracolo:	Restituito il documento creato.

DocumentService

Test:	DocumentService::findDocument() - OriginPlace: "Salerno" In una mock HTTP request.
Esiste:	Documento con: - ID: 1 - Name: "Registro delle nascite" - RetrievalDate: 2022-01-01 - RetrievalPlace: "Salerno" - OriginDate: 2022-01-01 - OriginPlace: "Salerno" - Path: "/"
Oracolo:	La risposta HTTP contiene il documento con ID 1.

Test:	DocumentService::createDocument() - Name: "Registro delle nascite" - RetrievalDate: 2022-01-01 - RetrievalPlace: "Salerno" - OriginDate: 2022-01-01 - OriginPlace: "Salerno" - Path: "/" In una mock HTTP request.
Esiste:	-
Oracolo:	La risposta HTTP contiene il documento creato.

Unità: Sottosistema Tree Management**NodeController**

Test:	NodeController::getNodes(1)
Esiste:	Nodo con ownerId = 1
Oracolo:	Il nodo esistente con ownerId 1 viene restituito.

Test:	NodeController::getNodes(1)
Esiste:	-
Oracolo:	Nessun nodo viene restituito.

Test:	NodeController::createNode() <ul style="list-style-type: none"> - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE"
Esiste:	-
Oracolo:	Restituito il nodo creato.

Test:	NodeController::updateNode() <ul style="list-style-type: none"> - ID: 2 - Nodo con FirstName = "Luigi", BirthPlace = "Giffoni"
Esiste:	Nodo con: <ul style="list-style-type: none"> - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE"
Oracolo:	Il nodo con ID 2 ha ora FirstName = "Luigi" e BirthPlace = "Giffoni".

Test:	NodeController::deleteNode() - ID: 2
Esiste:	Nodo con: - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE"
Oracolo:	Restituito il nodo con ID 2 eliminato.
Test:	NodeController::bindDocument() - NodeID: 2 - DocumentID: 3
Esiste:	Nodo con ID: 2 Documento con ID: 3 Documento con ID: 4
Oracolo:	Tra i documenti associati al nodo con ID 2 risulta il documento con ID 3.
Test:	NodeController::unbindDocument() - ID: 2
Esiste:	Nodo con ID: 2 Documento con ID: 3
Oracolo:	Tra i documenti associati al nodo con ID 2 non risulta il documento con ID 3.

TreeService

Test:	TreeService::getNodes(1) In una mock HTTP request.
Esiste:	Nodo con ownerId = 1
Oracolo:	La risposta HTTP contiene il nodo esistente con ownerId 1.
Test:	TreeService::createNode() <ul style="list-style-type: none"> - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE" In una mock HTTP request.
Esiste:	-
Oracolo:	La risposta HTTP contiene il nodo creato.
Test:	TreeService::createNode() <ul style="list-style-type: none"> - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE" In una mock HTTP request.
Esiste:	E' richiesto il lancio di un Unknown Error.
Oracolo:	La risposta HTTP l'errore 500 - Server Error.
Test:	TreeService::updateNode() <ul style="list-style-type: none"> - ID: 2 - Nodo con FirstName = "Luigi", BirthPlace = "Giffoni" In una mock HTTP request.
Esiste:	Nodo con: <ul style="list-style-type: none"> - ID: 2 - FirstName: "Mario"

	<ul style="list-style-type: none"> - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE"
Oracolo:	La risposta HTTP contiene il nodo con ID 2 con FirstName = "Luigi" e BirthPlace = "Giffoni".

Test:	TreeService::deleteNode() <ul style="list-style-type: none"> - ID: 2
Esiste:	Nodo con: <ul style="list-style-type: none"> - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE" In una mock HTTP request.
Oracolo:	La risposta HTTP contiene il nodo con ID 2 eliminato.

Test:	TreeService::bindDocument() <ul style="list-style-type: none"> - NodeID: 2 - DocumentID: 3 In una mock HTTP request.
Esiste:	Nodo con ID: 2 Documento con ID: 3 Documento con ID: 4
Oracolo:	La risposta HTTP contiene il nodo con ID 2: tra i documenti associati ad esso risulta il documento con ID 3.

Test:	TreeService::unbindDocument() <ul style="list-style-type: none"> - ID: 2 In una mock HTTP request.
Esiste:	Nodo con ID: 2 Documento con ID: 3
Oracolo:	La risposta HTTP contiene il nodo con ID 2: tra i documenti associati ad esso non risulta il documento con ID 3.

Unit Testing (server-side): test report**PASS** dist/account/controllers/account.controller.test.js**Account Controller**

- ✓ Should retrieve an existing username (1 ms)
- ✓ Should not retrieve an existing username (1 ms)
- ✓ Should create an account (2 ms)

PASS dist/account/services/auth.service.test.js**Authorization Service**

- ✓ Should authenticate a valid account (68 ms)
- ✓ Should not authenticate an account with non valid password (66 ms)
- ✓ Should not authenticate an account with non valid username (2 ms)
- ✓ Should not authenticate an account with empty username (1 ms)
- ✓ Should not authenticate an account with empty password (2 ms)
- ✓ Should not authenticate an account with empty password (2 ms)

PASS dist/account/services/registration.service.test.js**Registration Service**

- ✓ Should register an account (70 ms)
- ✓ Should register an account (65 ms)

PASS dist/tree/controllers/node.controller.test.js**Node Controller**

- ✓ Should find a list of nodes filtered by owner (4 ms)
- ✓ Should create a node (3 ms)
- ✓ Should update a node (4 ms)
- ✓ Should delete a node (2 ms)
- ✓ Should bind a document to a node (3 ms)
- ✓ Should unbind a document from a node (7 ms)

PASS dist/tree/services/tree.service.test.js**Tree Service**

- ✓ Should find a list of nodes filtered by owner (6 ms)
- ✓ Should create a node (2 ms)
- ✓ Should update a node (4 ms)
- ✓ Should handle any unknown error during creation (3 ms)
- ✓ Should delete a node (3 ms)
- ✓ Should bind a document to a node (3 ms)
- ✓ Should unbind a document from a node (4 ms)








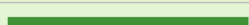

PASS dist/document/controllers/document.controller.test.js**Document Controller**

- ✓ Should find a list of existing documents (1 ms)
- ✓ Should create a document (1 ms)

PASS dist/document/services/document.service.test.js**Document Service**

- ✓ It should find a list of document (5 ms)
- ✓ It should create a document (2 ms)

Unit Testing (Server-side): coverage report

File ▲		Statements ⇅		Branches ⇅		Functions ⇅		Lines ⇅	
account/controllers/src/account/controllers		100%	13/13	100%	2/2	100%	3/3	100%	10/10
account/services/src/account/services		100%	36/36	100%	7/7	100%	4/4	100%	32/32
document/controllers/src/document/controllers		100%	13/13	100%	2/2	100%	3/3	100%	10/10
document/services/src/document/services		100%	16/16	100%	0/0	100%	3/3	100%	13/13
document/utls/src/document/utls		100%	7/7	75%	12/16	100%	2/2	100%	6/6
tree/controllers/src/tree/controllers		100%	29/29	100%	6/6	100%	7/7	100%	22/22
tree/models/src/tree/models		100%	17/17	100%	0/0	100%	3/3	100%	15/15
tree/services/src/tree/services		100%	40/40	100%	0/0	100%	7/7	100%	33/33
utls/src/utls		100%	7/7	100%	0/0	100%	2/2	100%	6/6

Report del testing della REST API

Per consultare i test eseguiti, riferirsi al Test Plan, sezione 9.2, la quale contiene inoltre gli identificativi associati a ciascun caso di test. Ulteriori endpoint non considerati da Category Partition sono esplicitati separatamente.

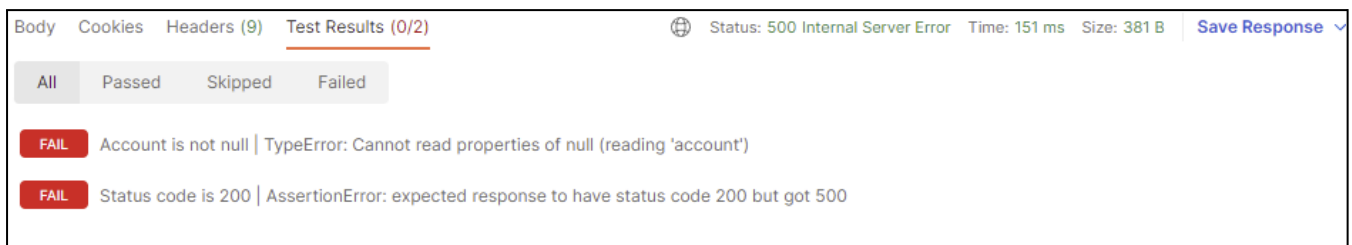
Report AccountRoute:

Incident Report: "BirthDate" is undefined - Test case: R-R-TC1

La data di nascita all'atto dell'inserimento di un nodo non viene inserita affatto. Dopo un'analisi del codice di AccountController, si riconduce il problema a

- La mancata inizializzazione della data nel metodo `createNode()`;
- Un (banale) errore di spelling tra i campi della richiesta HTTP simulata mediante Postman (i.e. "birtDate" anziché "birthDate").

Si comunica che è stato effettuato regression testing della REST API, in particolare dei test cases di registrazione in seguito all'incident ivi descritto.



In seguito alla correzione ed al regression testing, ogni test si è concluso con un PASS:

Report TreeRoute:

- GetNodes: l'endpoint a cui rivolgersi per ottenere la lista dei nodi.

Report DocumentRoute: