



# Università degli Studi di Salerno

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

## Quiad Unit Test Cases Specification

Versione 1.3  
30/12/2022



**QUIAD**  
Family Tree

**Partecipanti al progetto e scriventi**

Nome	Matricola
Di Pasquale Valerio	0512110638
Troisi Vito	0512109807

---

**Revision History**

Data	Versione	Descrizione	Autore
29/12/2022	1.0	Prima stesura, Indice, spostati qui i test di unità server-side	T.V.
29/12/2022	1.1	Rimosso paragrafo introduttivo non necessario	T.V.
30/12/2022	1.2	TCS per classi client dell'application domain	T.V.
30/12/2022	1.3	TCS per classi client: Angular Services	D.P.V. T.V.

**Indice**

1. Overview.....	p.4
2. Riferimenti.....	p.4
3. Specifica dei casi di test di unità	
3.1 Server-side.....	p.5
3.2 Client-side.....	p.15

## Overview

Il presente documento funge da appendice al piano di test della Web Application Quiad. In esso sono state presentate le specifiche dei casi di test strutturati per la REST API e per il sistema nella sua interezza. Al fine di evitare di rendere il Test Plan eccessivamente verboso, si è pensato di porre in un documento separato le specifiche dei casi di test di unità.

Per i casi suddetti, si osservi che a ciascuno è associato uno scope univoco, del tipo "Classe:Scope" che lo identificherà sul report dei test.

Saranno inoltre effettuati i test di creazione delle componenti grafiche Angular (detti "Component" o "Pages", termini non identici, ma equiparabili nel contesto di tali test).

## Riferimenti

Per procedere al meglio nella lettura del presente documento, si segnalano:

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

Naturalmente, in virtù di quanto affermato nel paragrafo introduttivo, il piano di test ricopre un ruolo senz'altro fondamentale per quanto concerne la comprensione dei casi test che saranno presentati.

**Specifica dei casi di test di unità: server-side****Unità: Sottosistema Account Management**

<b>AccountController</b>	
<b>Scope:</b>	AccountController:Should retrieve an existing username
<b>Test:</b>	AccountController::findByUsername("quiad")
<b>Esiste:</b>	Account con username = "quiad"
<b>Oracolo:</b>	Restituito account esistente con username "quiad".
<b>Scope:</b>	AccountController:Should not retrieve a non existing username
<b>Test:</b>	AccountController::findByUsername("quiadz")
<b>Esiste:</b>	Account con username = "quiad"
<b>Oracolo:</b>	Nessun account restituito.
<b>Scope:</b>	AccountController:Should create an account
<b>Test:</b>	AccountController::createAccount() - Username: "quiad" - Email: "quiad@test.com" - Password: "quiad"
<b>Esiste:</b>	-
<b>Oracolo:</b>	Restituito l'account creato.

**AuthService**

<b>Scope:</b>	AuthService:Should authenticate a valid account
<b>Test:</b>	AuthService::login() - Username: "quiad" - Password: "miapassword" In una mock HTTP request.
<b>Esiste:</b>	Account con username = "quiad"
<b>Oracolo:</b>	La risposta HTTP contiene l'utente il cui username è quiad.

  

<b>Scope:</b>	AuthService:Should not authenticate an account with non valid password
<b>Test:</b>	AuthService::login() - Username: "quiad" - Password: "miapasswordz" In una mock HTTP request.
<b>Esiste:</b>	Account con username = "quiad"
<b>Oracolo:</b>	La risposta HTTP contiene l'errore 401 - Unauthorized.

  

<b>Scope:</b>	AuthService:Should not authenticate an account with non valid username
<b>Test:</b>	AuthService::login() - Username: "daqh" - Password: "miapassword" In una mock HTTP request.
<b>Esiste:</b>	Account con username = "quiad"
<b>Oracolo:</b>	La risposta HTTP contiene l'errore 401 - Unauthorized.

  

<b>Scope:</b>	AuthService:Should not authenticate an account with empty username
<b>Test:</b>	AuthService::login() - Username: "" - Password: "miapassword" In una mock HTTP request.
<b>Esiste:</b>	Account con username = "daqh"

<b>Oracolo:</b>	La risposta HTTP contiene l'errore 400 - Bad Request.
-----------------	---

  

<b>Scope:</b>	AuthService:Should not authenticate an account with empty password
<b>Test:</b>	AuthService::login() - Username: "daqh" - Password: "" In una mock HTTP request.
<b>Esiste:</b>	Account con username = "daqh"
<b>Oracolo:</b>	La risposta HTTP contiene l'errore 400 - Bad Request.

  

<b>Scope:</b>	AuthService:Should not authenticate an account with empty password
<b>Test:</b>	AuthService::login() - Username: "daqh" - Password: "" In una mock HTTP request.
<b>Esiste:</b>	Il mock object PrismaClient è istanziato al fine di restituire un errore, simulando un DB non raggiungibile.
<b>Oracolo:</b>	La risposta HTTP contiene l'errore 500 - Server Error.

**RegistrationService**

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

<b>Scope:</b>	RegistrationService:Should not register an account
<b>Test:</b>	RegistrationService::register() <ul style="list-style-type: none"><li>- Email: "valeriotroisi@quiad.com"</li><li>- Username: "quiad"</li><li>- Password: "miapassword"</li></ul> In una mock HTTP request.
<b>Esiste:</b>	Account con username = "quiad"
<b>Oracolo:</b>	La risposta HTTP contiene l'errore 500 - Server Error.



**Unità: Sottosistema Document Management****DocumentController**

<b>Scope:</b>	DocumentController:Should find a list of existing documents
<b>Test:</b>	DocumentController::findDocument() - OriginPlace: "Salerno"
<b>Esiste:</b>	Documento con: <ul style="list-style-type: none"><li>- Name: "Registro delle nascite"</li><li>- RetrievalDate: 2022-01-01</li><li>- RetrievalPlace: "Salerno"</li><li>- OriginDate: 2022-01-01</li><li>- OriginPlace: "Salerno"</li><li>- Path: "/"</li></ul>
<b>Oracolo:</b>	Restituito il documento esistente.

**DocumentService**

<b>Scope:</b>	DocumentService:Should find a list of documents
<b>Test:</b>	DocumentService::findDocument() - OriginPlace: "Salerno" In una mock HTTP request.
<b>Esiste:</b>	Documento con: - ID: 1 - Name: "Registro delle nascite" - RetrievalDate: 2022-01-01 - RetrievalPlace: "Salerno" - OriginDate: 2022-01-01 - OriginPlace: "Salerno" - Path: "/"
<b>Oracolo:</b>	La risposta HTTP contiene il documento con ID 1.

**Unità: Sottosistema Tree Management****NodeController**

<b>Scope:</b>	NodeController:Should find a list of nodes filtered by owner
<b>Test:</b>	NodeController::getNodes(1)
<b>Esiste:</b>	Nodo con ownerId = 1
<b>Oracolo:</b>	Il nodo esistente con ownerId 1 viene restituito.

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

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

<b>Scope:</b>	NodeController:Should delete a node
<b>Test:</b>	NodeController::deleteNode() - ID: 2
<b>Esiste:</b>	Nodo con: - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE"
<b>Oracolo:</b>	Restituito il nodo con ID 2 eliminato.

  

<b>Scope:</b>	NodeController:Should bind a document to a node
<b>Test:</b>	NodeController::bindDocument() - NodeID: 2 - DocumentID: 3
<b>Esiste:</b>	Nodo con ID: 2 Documento con ID: 3 Documento con ID: 4
<b>Oracolo:</b>	Tra i documenti associati al nodo con ID 2 risulta il documento con ID 3.

  

<b>Scope:</b>	NodeController:Should unbind a document from a node
<b>Test:</b>	NodeController::unbindDocument() - ID: 2
<b>Esiste:</b>	Nodo con ID: 2 Documento con ID: 3, associato al nodo con ID = 2
<b>Oracolo:</b>	Tra i documenti associati al nodo con ID 2 non risulta il documento con ID 3.

### TreeService

<b>Scope:</b>	TreeService:Should find a list of nodes filtered by owner
<b>Test:</b>	TreeService::getNodes(1) In una mock HTTP request.
<b>Esiste:</b>	Nodo con ownerId = 1
<b>Oracolo:</b>	La risposta HTTP contiene il nodo esistente con ownerId 1.

<b>Scope:</b>	TreeService:Should create a node
<b>Test:</b>	TreeService::createNode() <ul style="list-style-type: none"> <li>- ID: 2</li> <li>- FirstName: "Mario"</li> <li>- LastName: "Rossi"</li> <li>- BirthDate: 1990-02-01</li> <li>- BirthPlace: "Salerno"</li> <li>- OwnerID: 1</li> <li>- Sex: "MALE"</li> </ul> In una mock HTTP request.
<b>Esiste:</b>	-
<b>Oracolo:</b>	La risposta HTTP contiene il nodo creato.

<b>Scope:</b>	TreeService:Should handle any unknown error during creation
<b>Test:</b>	TreeService::createNode() <ul style="list-style-type: none"> <li>- ID: 2</li> <li>- FirstName: "Mario"</li> <li>- LastName: "Rossi"</li> <li>- BirthDate: 1990-02-01</li> <li>- BirthPlace: "Salerno"</li> <li>- OwnerID: 1</li> <li>- Sex: "MALE"</li> </ul> In una mock HTTP request.
<b>Esiste:</b>	E' richiesto il lancio di un Unknown Error.
<b>Oracolo:</b>	La risposta HTTP l'errore 500 - Server Error.

<b>Scope:</b>	TreeService:Should update a node
<b>Test:</b>	TreeService::updateNode() - ID: 2 - Nodo con FirstName = "Luigi", BirthPlace = "Giffoni" In una mock HTTP request.
<b>Esiste:</b>	Nodo con: - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE"
<b>Oracolo:</b>	La risposta HTTP contiene il nodo con ID 2 con FirstName = "Luigi" e BirthPlace = "Giffoni".

<b>Scope:</b>	TreeService:Should delete a node
<b>Test:</b>	TreeService::deleteNode() - ID: 2 In una mock HTTP request.
<b>Esiste:</b>	Nodo con: - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - BirthDate: 1990-02-01 - BirthPlace: "Salerno" - OwnerID: 1 - Sex: "MALE" In una mock HTTP request.
<b>Oracolo:</b>	La risposta HTTP contiene il nodo con ID 2 eliminato.

<b>Scope:</b>	TreeService:Should bind a document to a node
<b>Test:</b>	TreeService::bindDocument() - NodeID: 2 - DocumentID: 3 In una mock HTTP request.
<b>Esiste:</b>	Nodo con ID: 2 Documento con ID: 3 Documento con ID: 4
<b>Oracolo:</b>	La risposta HTTP contiene il nodo con ID 2: tra i documenti

	associati ad esso risulta il documento con ID 3.
<b>Scope:</b>	TreeService:Should unbind a document from a node
<b>Test:</b>	TreeService::unbindDocument() - ID: 2 In una mock HTTP request.
<b>Esiste:</b>	Nodo con ID: 2 Documento con ID: 3, associato al nodo con ID = 2
<b>Oracolo:</b>	La risposta HTTP contiene il nodo con ID 2: tra i documenti associati ad esso non risulta il documento con ID 3.

**Specifica dei test di unità: client-side**

Per le classi dell'Application Domain:

**UtenteBase**

<b>Scope:</b>	UtenteBase:Should insert a node in user tree
<b>Test:</b>	Item: l'utente su cui viene invocato il metodo UtenteBase::addNode() - ID: 2
<b>Esiste:</b>	-
<b>Oracolo:</b>	La lista di nodi dell'utente contiene il nodo con ID = 2

<b>Scope:</b>	UtenteBase:Should modify a certain node in the user tree
<b>Test:</b>	UtenteBase::modifyNode() - ID = 2 - Nodo con FirstName = "Gennaro"
<b>Esiste:</b>	Item: l'utente su cui viene invocato il metodo Nodo con: - ID: 2 - FirstName: ""
<b>Oracolo:</b>	La lista di nodi dell'utente il nodo con ID 2 con FirstName = "Gennaro".

<b>Scope:</b>	UtenteBase:Should delete a certain node in the user tree
<b>Test:</b>	UtenteBase::deleteNode() - ID: 2
<b>Esiste:</b>	Item: l'utente su cui viene invocato il metodo Nodo con: - ID: 2
<b>Oracolo:</b>	La lista di nodi dell'utente non contiene il nodo con ID 2.



### Node

<b>Scope:</b>	Node:Should bind a document to a node
<b>Test:</b>	Node::bindDocument() - DocumentID: 1
<b>Esiste:</b>	Item: il nodo su cui viene invocato il metodo Documento con: - ID: 1 - Name: "Registro delle nascite" - RetrievalDate: "2009-12-09" - RetrievalPlace: "Salerno" - OriginDate: "1990-12-09" - OriginPlace: "Battipaglia"
<b>Oracolo:</b>	La lista dei documenti associati al nodo contiene il documento con ID 1.

<b>Scope:</b>	Node:Should unbind a document from a node
<b>Test:</b>	Node::unbindDocument()
<b>Esiste:</b>	Item: il nodo su cui viene invocato il metodo Documento con ID: 3, associato al nodo
<b>Oracolo:</b>	La lista dei documenti associati al nodo non contiene il documento con ID 1.

Si osservi che ciascun oggetto è univocamente rappresentato dal suo ID.

Per Angular Services:

### LoginService

<b>Scope:</b>	LoginService:Should be created
<b>Test:</b>	- service = inject(AuthService)
<b>Esiste:</b>	-
<b>Oracolo:</b>	Service risulta true, i.e. correttamente creato.
<b>Scope:</b>	LoginService:Should login and remember the user
<b>Test:</b>	AuthService::login() - Username: "quiad" - Password: "miapassword" - RememberMe: true In una mock HTTP request.
<b>Esiste:</b>	Account con username = "quiad"
<b>Oracolo:</b>	La risposta HTTP contiene l'utente il cui username è quiad e il token di autenticazione.
<b>Scope:</b>	LoginService:Should login and not remember the user
<b>Test:</b>	AuthService::login() - Username: "quiad" - Password: "miapassword" - RememberMe: false In una mock HTTP request.
<b>Esiste:</b>	Account con username = "quiad"
<b>Oracolo:</b>	La risposta HTTP contiene l'utente il cui username è quiad.
<b>Scope:</b>	LoginService:Should not login with a non valid password
<b>Test:</b>	AuthService::login() - Username: "quiad" - Password: "miapasswordz" In una mock HTTP request.
<b>Esiste:</b>	Account con username = "quiad"

<b>Oracolo:</b>	La risposta HTTP contiene l'errore 401 - Unauthorized.
<b>Scope:</b>	LoginService:Should not login with a non valid username
<b>Test:</b>	AuthService::login() - Username: "daqh" - Password: "miapassword" In una mock HTTP request.
<b>Esiste:</b>	Account con username = "quiad"
<b>Oracolo:</b>	La risposta HTTP contiene l'errore 401 - Unauthorized.

### LogoutService

<b>Scope:</b>	LogoutService:Should be created
<b>Test:</b>	- service = inject(LogoutService)
<b>Esiste:</b>	-
<b>Oracolo:</b>	Service risulta true, i.e. correttamente creato.
<b>Scope:</b>	LogoutService:Should logout
<b>Test:</b>	LogoutService::logout()
<b>Esiste:</b>	Account con username = "quiad" autenticato ed il suo token di autenticazione. E' simulato che tale utente si trovi alla homepage.
<b>Oracolo:</b>	Il token di autenticazione dell'account con username = "quiad" non è presente.

### RegistrationService

<b>Scope:</b>	RegistrationService:Should be created
<b>Test:</b>	- service = inject(RegistrationService)
<b>Esiste:</b>	-
<b>Oracolo:</b>	Service risulta true, i.e. correttamente creato.

<b>Scope:</b>	RegistrationService:Should register a new user
<b>Test:</b>	RegistrationService::register() - Account con: Email: "valeriotroisi@quiad.com" Username: "quiad" Password: "miapassword" In una mock HTTP request.
<b>Esiste:</b>	-
<b>Oracolo:</b>	La risposta HTTP contiene l'utente registrato con username = "quiad".

### DocumentService

<b>Scope:</b>	DocumentService:Should be created
<b>Test:</b>	- service = inject(DocumentService)
<b>Esiste:</b>	-
<b>Oracolo:</b>	Service risulta true, i.e. correttamente creato.

<b>Scope:</b>	DocumentService:Should find a list of documents
<b>Test:</b>	DocumentService::findDocument() - OriginPlace: "Salerno" In una mock HTTP request.
<b>Esiste:</b>	Documento con: <ul style="list-style-type: none"> <li>- ID: 1</li> <li>- Name: "Registro nascite"</li> <li>- CategoryID: 1</li> <li>- RetrievalDate: 2022-01-01</li> <li>- RetrievalPlace: "Salerno"</li> <li>- OriginDate: 2022-01-01</li> <li>- OriginPlace: "Salerno"</li> <li>- Path: "/"</li> </ul>
<b>Oracolo:</b>	La risposta HTTP contiene il documento con ID 1.

### TreeService

<b>Scope:</b>	TreeService:Should be created
<b>Test:</b>	- service = inject(TreeService)
<b>Esiste:</b>	-
<b>Oracolo:</b>	Service risulta true, i.e. correttamente creato.

<b>Scope:</b>	TreeService:Should get a list of nodes by ownerId
<b>Test:</b>	TreeService::getNodes(1) In una mock HTTP request.
<b>Esiste:</b>	Nodo con ID = 2 e ownerId = 1
<b>Oracolo:</b>	La risposta HTTP contiene una lista con il nodo esistente il quale ha ID = 2 e ownerId = 1.

<b>Scope:</b>	TreeService:Should create a certain node
<b>Test:</b>	TreeService::createNode() - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - OwnerID: 1 In una mock HTTP request.
<b>Esiste:</b>	-
<b>Oracolo:</b>	La risposta HTTP contiene il nodo creato.

<b>Scope:</b>	TreeService:Should modify a certain node
<b>Test:</b>	TreeService::updateNode() - ID: 2 - Nodo con FirstName = "Mario". In una mock HTTP request.
<b>Esiste:</b>	Nodo con: - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - OwnerID: 1

<b>Oracolo:</b>	La risposta HTTP contiene il nodo modificato.
-----------------	---

  

<b>Scope:</b>	TreeService:Should delete a certain node in the user tree
<b>Test:</b>	TreeService::deleteNode() - ID: 2 In una mock HTTP request.
<b>Esiste:</b>	Nodo con: - ID: 2 - FirstName: "Mario" - LastName: "Rossi" - OwnerID: 1
<b>Oracolo:</b>	La risposta HTTP contiene il nodo con ID 2 eliminato.

  

<b>Scope:</b>	TreeService:Should bind a document to a node
<b>Test:</b>	TreeService::bindDocument() - NodeID: 2 - DocumentID: 1 In una mock HTTP request.
<b>Esiste:</b>	Nodo con ID: 2 Documento con ID: 1
<b>Oracolo:</b>	La risposta HTTP contiene il nodo con ID 2: tra i documenti associati ad esso risulta il documento con ID 1.

  

<b>Scope:</b>	TreeService:Should unbind a document from a node
<b>Test:</b>	TreeService::unbindDocument() - NodeID: 2 - DocumentID: 3 In una mock HTTP request.
<b>Esiste:</b>	Nodo con ID: 2 Documento con ID: 3, associato al nodo con ID = 2
<b>Oracolo:</b>	La risposta HTTP contiene il nodo con ID 2: tra i documenti associati ad esso non risulta il documento con ID 3.

## Per Angular Components:

**DashboardPage**

<b>Scope:</b>	DashboardPage:Should create
<b>Test:</b>	Creazione del componente c:DashboardPage
<b>Esiste:</b>	-
<b>Oracolo:</b>	L'oggetto c, convertito in booleano, risulta true, il che segnala la sua avvenuta creazione.



