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 1.4
30/12/2022

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.
30/12/2022	1.4	TCS per classi client: Angular Components	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.17

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

AccountController

Scope:	AccountController:Should retrieve an existing username
Test:	AccountController::findByUsername(“quiad”)
Esiste:	Account con username = “quiad”
Oracolo:	Restituito account esistente con username “quiad”.

Scope:	AccountController:Should not retrieve a non existing username
Test:	AccountController::findByUsername(“quiadz”)
Esiste:	Account con username = “quiad”
Oracolo:	Nessun account restituito.

Scope:	AccountController:Should create an account
Test:	AccountController::createAccount() - Username: “quiad” - Email: “quiad@test.com” - Password: “quiad”
Esiste:	-
Oracolo:	Restituito l’account creato.

AuthService

Scope:	AuthService:Should authenticate a valid account
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.

Scope:	AuthService:Should not authenticate an account with non valid password
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.

Scope:	AuthService:Should not authenticate an account with non valid username
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.

Scope:	AuthService:Should not authenticate an account with empty username
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.

Scope:	AuthService:Should not authenticate an account with empty password
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.

Scope:	AuthService:Should not authenticate an account with empty password
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	
Scope:	RegistrationService:Should register an account
Test:	RegistrationService::register() - Email: “valeriotroisi@quiad.com” - Username: “quiad” - Password: “miapassword” In una mock HTTP request.
Esiste:	-
Oracolo:	La risposta HTTP contiene l’utente registrato.

Scope:	RegistrationService:Should not register an account
Test:	RegistrationService::register() - 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	
Scope:	DocumentController:Should find a list of existing documents
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.

DocumentService	
Scope:	DocumentService:Should find a list of documents
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.

Unità: Sottosistema Tree Management

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

Scope:	NodeController:Should create a node
Test:	NodeController::createNode() - ID: 2 - FirstName: “Mario” - LastName: “Rossi” - BirthDate: 1990-02-01 - BirthPlace: “Salerno” - OwnerID: 1 - Sex: “MALE”
Esiste:	-
Oracolo:	Restituito il nodo creato.

Scope:	NodeController:Should update a node
Test:	NodeController::updateNode() - ID: 2 - Nodo con FirstName = “Luigi”, BirthPlace = “Giffoni”
Esiste:	Nodo con: - 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”.

Scope:	NodeController:Should delete a node
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.

Scope:	NodeController:Should bind a document to a node
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.

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

TreeService

Scope:	TreeService:Should find a list of nodes filtered by owner
Test:	TreeService::getNodes(2) In una mock HTTP request.
Esiste:	Nodo con ownerID = 2
Oracolo:	La risposta HTTP contiene il nodo esistente con ownerID = 2.

Scope:	TreeService:Should not find a list of node belonging to another user
Test:	TreeService::getNodes(2) In una mock HTTP request.
Esiste:	- Utente con ID = 1, che effettua la richiesta - Utente con ID = 2
Oracolo:	La risposta HTTP contiene l'errore 403 - Forbidden.

Scope:	TreeService:Should create a node
Test:	TreeService::createNode() - 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.

Scope:	TreeService:Should handle any unknown error during creation
Test:	TreeService::createNode() - 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.

Scope:	TreeService:Should update a node
Test:	TreeService::updateNode() - ID: 2 - Nodo con FirstName = “Luigi”, BirthPlace = “Giffoni” In una mock HTTP request.
Esiste:	Nodo con: - ID: 2 - FirstName: “Mario” - 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”.

Scope:	TreeService:Should delete a node
Test:	TreeService::deleteNode() - ID: 2 In una mock HTTP request.
Esiste:	Nodo con: - 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.

Scope:	TreeService:Should not delete the user node
Test:	TreeService::deleteNode() - ID: 1 In una mock HTTP request.
Esiste:	Nodo utente con ID = 1
Oracolo:	La risposta HTTP contiene l'errore 500 - Server Error.

Scope:	TreeService:Should not delete a node that doesn't exist
Test:	TreeService::deleteNode() - ID: 2 In una mock HTTP request.
Esiste:	-
Oracolo:	La risposta HTTP contiene l'errore 500 - Server Error.

Scope:	TreeService:Should bind a document to a node
Test:	TreeService::bindDocument() - 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.

Scope:	TreeService:Should not perform bind if either the node or the document doesn't exist
Test:	TreeService::bindDocument() - NodeID: 2 - DocumentID: 3 In una mock HTTP request.
Esiste:	Nodo con ID: 2 Documento con ID: 3
Oracolo:	La risposta HTTP contiene l'errore 500 - Server Error.

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

Scope:	TreeService:Should not perform unbind if either the node or the document doesn’t exist
Test:	TreeService::bindDocument() - 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 l’errore 500 - Server Error.

Scope:	TreeService:Should throw an error when unbinding non-bound document from a node
Test:	NodeController::unbindDocument() - ID: 4
Esiste:	Nodo con ID: 2 Documento con ID: 4, associato al nodo con ID = 2
Oracolo:	Viene lanciata l’eccezione Prisma.UnknownRequestError: “The document doesn’t exist”

Specifica dei test di unità: client-side

Per le classi dell'Application Domain:

UtenteBase

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

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

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

Node	
Scope:	Node:Should bind a document to a node
Test:	Node::bindDocument() - DocumentID: 1
Esiste:	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”
Oracolo:	La lista dei documenti associati al nodo contiene il documento con ID 1.

Scope:	Node:Should unbind a document from a node
Test:	Node::unbindDocument()
Esiste:	Item: il nodo su cui viene invocato il metodo Documento con ID: 3, associato al nodo
Oracolo:	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

Scope:	LoginService:Should be created
Test:	- service = inject(AuthService)
Esiste:	-
Oracolo:	Service risulta true, i.e. correttamente creato.

Scope:	LoginService:Should login and remember the user
Test:	AuthService::login() - Username: “quiad” - Password: “miapassword” - RememberMe: true In una mock HTTP request.
Esiste:	Account con username = “quiad”
Oracolo:	La risposta HTTP contiene l’utente il cui username è quiad e il token di autenticazione.

Scope:	LoginService:Should login and not remember the user
Test:	AuthService::login() - Username: “quiad” - Password: “miapassword” - RememberMe: false In una mock HTTP request.
Esiste:	Account con username = “quiad”
Oracolo:	La risposta HTTP contiene l’utente il cui username è quiad.

Scope:	LoginService:Should not login with a non valid password
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.

Scope:	LoginService:Should not login with a non valid username
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.

LogoutService

Scope:	LogoutService:Should be created
Test:	- service = inject(LogoutService)
Esiste:	-
Oracolo:	Service risulta true, i.e. correttamente creato.

Scope:	LogoutService:Should logout
Test:	LogoutService::logout()
Esiste:	Account con username = “quiad” autenticato ed il suo token di autenticazione. E’ simulato che tale utente si trovi alla homepage.
Oracolo:	Il token di autenticazione dell’account con username = “quiad” non è presente.

RegistrationService

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

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

DocumentService

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

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

TreeService

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

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

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

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

Oracolo:	La risposta HTTP contiene il nodo modificato.

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

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

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

Per gli Angular Components:

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

Il test per ciascun AngularComponent è identico e ogni component viene testato: si è ritenuto pertanto superfluo riportare ogni test dallo scope ":Should create" per i component rendendo così eccessivamente verboso il presente documento.