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 1.1
29/12/2022

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.

Indice

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.

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() - 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

Scope:	TreeService:Should find a list of nodes filtered by owner
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.

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 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 unbind a document from a node
Test:	TreeService::unbindDocument() - 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.

Specifica dei test di unità: client-side