0.1 Data Layer

Le API del data layer sono state testate tramite JUnit5. E' stato generato un test case per ogni API presente nel livello; i casi di test hanno solo testato il funzionamento delle funzioni in modo grossolano. Prima di progettare in modo più dettagliato i casi di test, sarebbe opportuno aumentare la robustezza del software ad esempio implementando un'attenta validazione dei parametri di ingresso delle funzioni.

Nella seguente figura è mostrato un esempio del caso di test implementato con JUnit5 per l'API di Utente.

```
/**
 * @author PsssTeam
 *Il test richiede che nel db vi sia una tupla in Utente con password = "testPassword" e idUser = 1
 */
class TestUtente {
    private static ControllerUtenteDB controller;



    /**
     * @throws java.lang.Exception
     */
    @BeforeAll
    static void setUpBeforeClass() throws Exception {
        controller = new ControllerUtenteDB();


    }

    /**
     * Test method for {@link dataLayer.user.controller.ControllerUtenteDB#validateUser(dataLayer.utilities.idUser)}.
     */
    @Test
    void testValidateUser() {
        if (controller.validateUser(new idUser(1))==StateResult.VALID) {
            System.out.println("testValidateUser: Output VALID verificato");
        }
    }

    /**
     * Test method for {@link dataLayer.user.controller.ControllerUtenteDB#createUser(dataLayer.user.entities.UtenteDB, java.lang.String)}.
     */
    @Test
    void testCreateUser() {

        UtenteDB utente = new UtenteDB("Giorgio", "Farina", "giorgio1996.fari96@gmail.com");
        if (controller.createUser(utente, "testPassword")==StateResult.CREATED) {
            System.out.println("createUser: Id Utente: Output CREATED "+utente.getId().toString()+"\n");

        };

    }

    /**
     * Test method for {@link dataLayer.user.controller.ControllerUtenteDB#retrieveUser(dataLayer.utilities.idUser, dataLayer.user.entities.UtenteDB)}.
     */
    @Test
    void testRetrieveUser() {

        UtenteDB utente = new UtenteDB();
        if (controller.retrieveUser(new idUser(1), utente)==StateResult.VALID) {
            System.out.println("retrieveUser: Output VALID "+utente.toString()+"\n");
        };
    }
```

Figura 1: Esempio di TestCase JUnit 5

0.2 REST API

Il paradigma architetturale REST è basato sul protocollo HTTP. Anche in questo caso non è stato progettato un vero e propio suite di casi di test per testare ogni risorsa REST, piuttosto, su Swagger, le API sono state documentate secondo lo standard OpenAPI3.0 ottenendo come risultato un ambiente di test, nel quale, come è possibile vedere dall'immagine seguente, ogni URI puà essere testata facilemente modificando i parametri di ingresso alla funzione. Inoltre ad ogni metodo di ogni URI è stato documentato uno schema per la risposta con il quale è possibile sia progettare i test, ma anche validare la risposta.

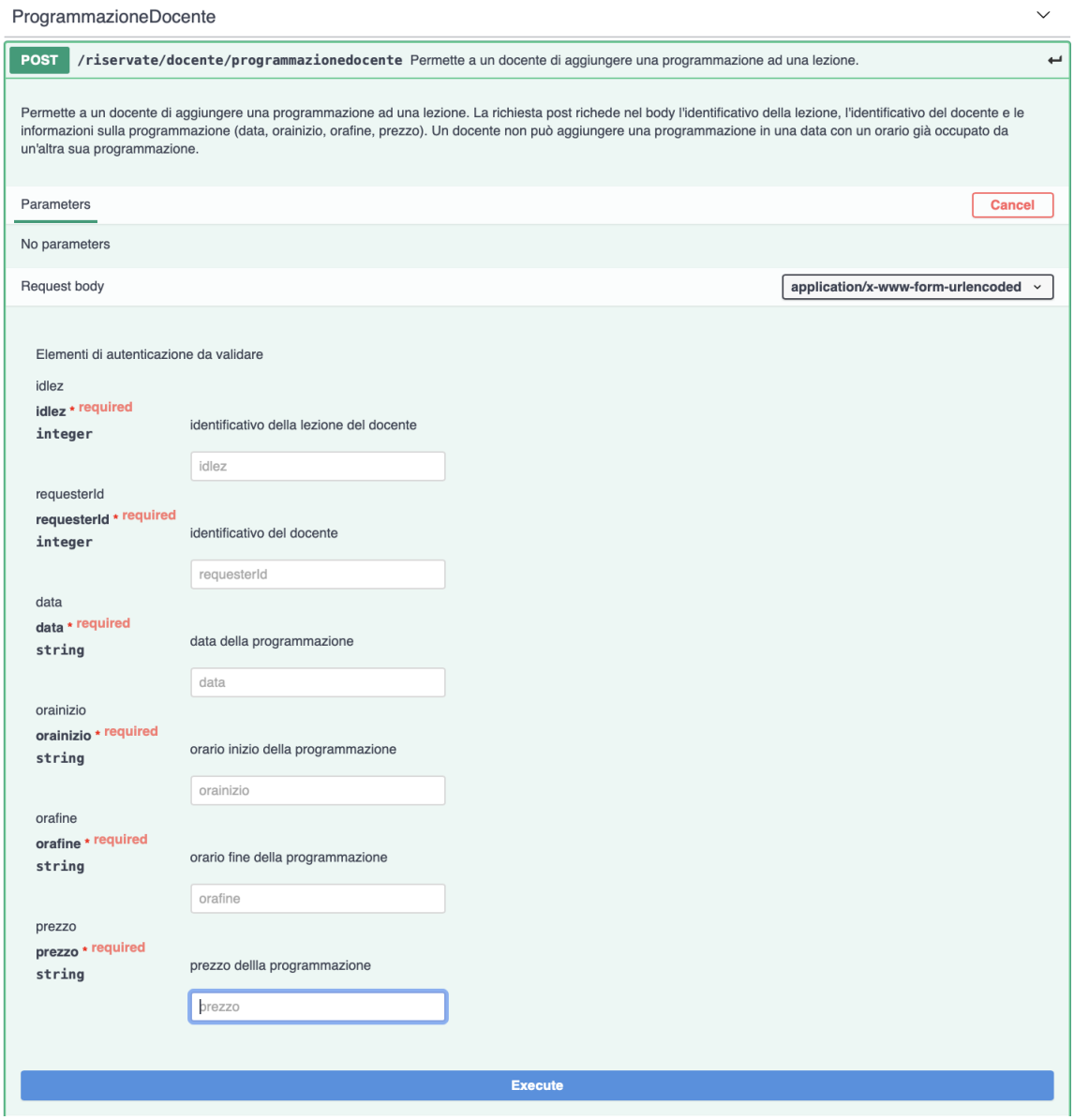

Figura 2: Esempio di Interfaccia Swagger per testare il metodo POST dell'URI "ProgrammazioneDocente"

Figura 3: Schema di risposta alla richiesta POST dell'URI "ProgrammazioneDocente"