FRAMEWORK DE TESTARE UNITARĂ DIN JAVA

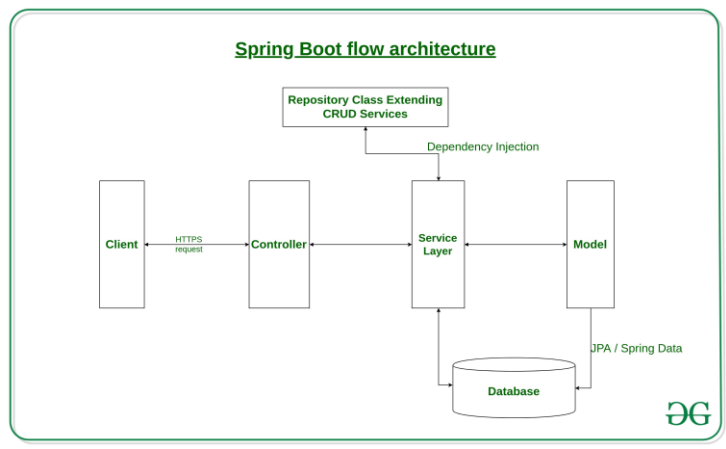
GRUPA 333
Diaconu Stefan-Mircea, Dina George-Alexandru, Enescu Irina-Stefania,
Mihaila Nicolae, Potanga Alexandru-Alin

SPRINGBOOT

Spring Boot reprezinta un framework pentru dezvoltarea aplicatiilor Java prin crearea unor API-uri. Proiectul initial pe baza caruia am ales sa ilustram strategiile de testare unitara in Java constituie serverul de backend al unei aplicatii web.

- Repository layer - Spring Data JPA
- Service layer
- Controller layer - cererile HTTP

Arhitectura proiectului respecta modelul din figura.



https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/01.-Prezentarea-proiectului-intial

JUNIT 5 – FRAMEWORK TESTARE

Pentru implementarea testelor unitare am utilizat **JUnit 5**, un framework de testare pentru limbajul de programare Java ce ofera flexibilitate, extensibilitate si usurinta in scrierea si rularea acestora. În cadrul dezvoltării am întâlnit următoarele noțiuni:

- **Mock** (@Mock, @InjectMocks) - o simulare a unui obiect real ce reproduce comportamentul si interactiunile cu alte componente intr-un mod controlat, facilitând izolarea si testarea corecta a componentei principale.
- **Assert** (assertThrows, assertEquals, assertNotEquals etc.) - in urma apelarii functiei pe care dorim sa o testam comparăm valoarea obținuta (actual) cu asteptarile noastre (expected) - in cazul in care valorile sunt egale, testul va trece, altfel acesta va esua.

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/02.-Implementarea-testelor-unitare-cu-JUnit-5

TOOL AI – ChatGPT 3.5

```java
@Test
public void getAuthenticatedPatientProfile_returnPatientProfile() {
    // Given
    var firstName = "John";
    var lastName = "Doe";
    var sex = Sex.MALE;
    var birthdate = LocalDate.now();
    var accountEmail = "john@example.com";
    var patient = new Patient(firstName, lastName, sex, birthdate);
    when(accountService.getAuthenticatedUserEmail()).thenReturn(accountEmail);
    when(patientRepository.findPatientProfileByEmail(any())).thenReturn(patient);

    // When
    var patientCreated = patientService.getAuthenticatedPatientProfile();

    // Then
    assertEquals(firstName, patientCreated.getFirstName());
    assertEquals(lastName, patientCreated.getLastName());
    assertEquals(sex, patientCreated.getSex());
    assertEquals(birthdate, patientCreated.getBirthdate());
}
```

TESTE PROPRII

- getAuthenticatedPatientProfile -

```java
@Test
public void getAuthenticatedPatientProfile_throwPatientProfileNotFoundException() {
    // Given
    var accountEmail = "john@example.com";
    when(accountService.getAuthenticatedUserEmail()).thenReturn(accountEmail);
    when(patientRepository.findPatientProfileByEmail(any())).thenReturn( t: null);

    // When
    // Then
    assertThrows(PatientProfileNotFoundException.class,
            () -> patientService.getAuthenticatedPatientProfile());
}
```

Testele au fost implementate pentru a acoperi atât cazul unui succes, cât și cazul unei erori.

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/03.-Raport-despre-folosirea-unui-tool-AI-%E2%80%90-chatGPT

TOOL AI – ChatGPT 3.5

```java
@Test
void testGetAuthenticatedPatientProfile() {
    // Mock behavior for patientRepository
    Patient mockPatient = new Patient(/* Initialize mock patient data */);
    when(patientRepository.findPatientProfileByEmail(anyString())).thenReturn(mockPatient);

    // Call the method under test
    Patient result = patientService.getAuthenticatedPatientProfile();

    // Verify that the method returned the expected result
    assertNotNull(result);
    assertEquals(mockPatient, result);
}

new *
@Test
void testGetAuthenticatedPatientProfile_whenProfileNotFound() {
    // Mock behavior for patientRepository to return null
    when(patientRepository.findPatientProfileByEmail(anyString())).thenReturn( t: null);

    // Verify that the method under test throws the expected exception
    assertThrows(PatientProfileNotFoundException.class, () -> patientService.getAuthenticatedPatientProfile());
}
```

TESTE GENERATE

- getAuthenticatedPatientProfile -

```java
new *
@BeforeEach
void setUp() {
    // Mock behavior for accountService
    when(accountService.getAuthenticatedUserEmail()).thenReturn( t: "test@example.com");
}
```

- Metoda setUp()
- Adnotarea @BeforeEach

Testele generate de chatGPT sunt asemanatoare si acopera ambele cazuri identificate anterior.

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/03.-Raport-despre-folosirea-unui-tool-AI-%E2%80%90-chatGPT

CONCLUZII

1. Metoda setUp() cu adnotarea @BeforeEach se apeleaza inaintea fiecarui test pentru a instantia mock-ul accountService, returnand se fiecare data acelasi raspuns. **Beneficiul** adus este reprezentat de reducerea duplicarii codului.

2. Instanțierea newPatient() – deoarece nu am oferit detalii referitoare la clasa Patient, obiectul mockPatient a fost creat cu un constructor fara parametrii, dar existenta acestora a fost luata in considerare.

3. Ca urmare a punctului 2, primul test utilizeaza assertNotNull() si assertEquals() pe obiecte intrucat nu au fost identificare campurile pe care sa faca assert.

AMBELE TESTE AU TRECUT CU SUCCES!!

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/03.-Raport-despre-folosirea-unui-tool-AI-%E2%80%90-chatGPT

TESTE INTEGRARE

Am implementat teste de integrare pentru a verifica cum funcționează împreună diferitele componente ale aplicației web. Aceste teste sunt esențiale pentru a asigura că modulele individuale, care funcționează corect în izolare, interacționează corespunzător atunci când sunt integrate.

```
@Test
void specializationsRequest_authenticated_returnsSpecializationsList() throws Exception {
    var account1 = accountService.createDoctorAccount( email: "userd1@example.com", password: "Passwordd1");
    var doctor1 = doctorService.createDoctorProfile( firstName: "Alex", lastName: "Doe", specialization: "Cardiology", account1.getEmail());

    var account2 = accountService.createDoctorAccount( email: "userd2@example.com", password: "Passwordd2");
    var doctor2 = doctorService.createDoctorProfile( firstName: "Sam", lastName: "Doe", specialization: "Neurology", account2.getEmail());

    var account3 = accountService.createDoctorAccount( email: "userd3@example.com", password: "Passwordd3");
    var doctor3 = doctorService.createDoctorProfile( firstName: "Rose", lastName: "Doe", specialization: "Dermatology", account3.getEmail());

    mockMvc.perform(requestTester.authenticatedGet( url: "/specializations"))
            .andExpect(status().isOk())
            .andExpect(jsonPath( expression: "$[*]", hasItem(doctor1.getSpecialization())))
            .andExpect(jsonPath( expression: "$[*]", hasItem(doctor2.getSpecialization())))
            .andExpect(jsonPath( expression: "$[*]", hasItem(doctor3.getSpecialization())));
}
```

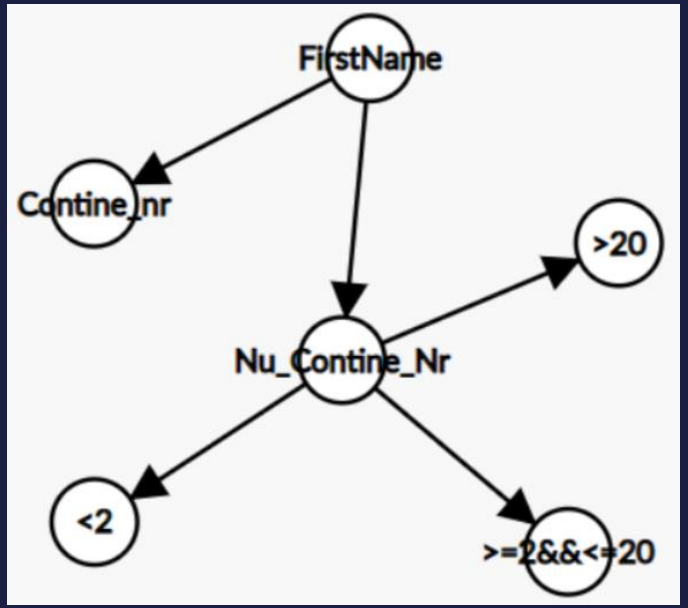
https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/04.-Implementarea-testelor-de-integrare

CLASE DE ECHIVALENTA

```
19    public class Patient extends Profile {
20        @Column(name = "first_name")  no usages
21        @Size(min = 2,max = 20, message = "First name cannot be shorter than 2 or longer than 20")
22        @Pattern(regexp = "^[^0-9]+$", message = "First name cannot contain numbers")
23        private String firstName;
```

Constrangerea @Size - limitarea lungimii numelor
Constrangerea @Pattern – fara cifre in nume

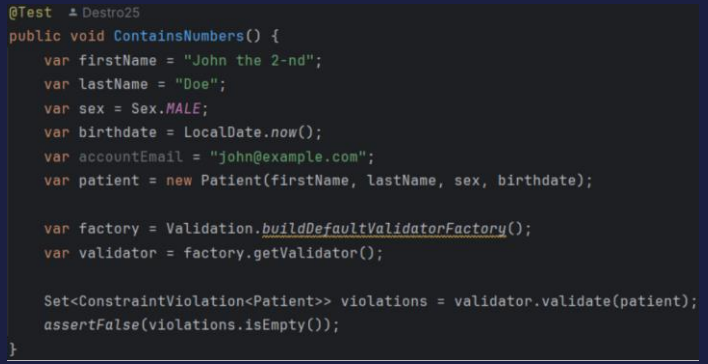
1. Clasa de echivalenta a string-urilor care contin numere
2. Clasa de echivalenta a string-urilor care nu contin numere
    a. Clasa de echivalenta a string-urilor cu o lungime < 2
    b. Clasa de echivalenta a string-urilor cu o lungime > 2 si < 20
    c. Clasa de echivalenta a string-urilor cu o lungime > 20



https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/05.-Partitionarea-in-clase-de-echivalenta

ANALIZA VALORI DE FRONTIERA

Impartirea in clase de echivalenta favorizeaza alegerea valorilor de la frontiera acestor clase.
Ca prim exemplu am ales sa implementam un test pe cazul in care string-ul nostru contine un numar sau mai multe.



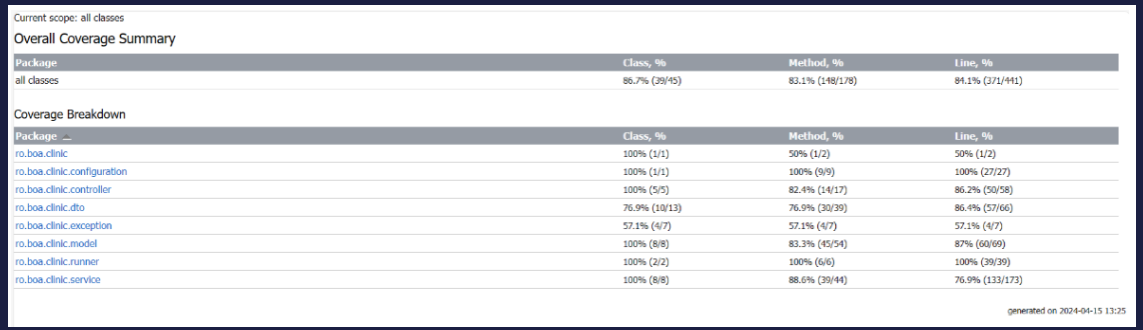
Am utilizat **buildDefaultValidatorFactory** si **getValidator** pentru a crea o instanta de Validator. Am folosit acesta instanta de validator pentru a verifica daca firstName verifica toate restrictiile din Patient. Daca acestea nu sunt verificate, vom retine mesajul constrangerii incalcate.

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/06.-Analiza-valorilor-de-frontiera

ACOPERIRE LA NIVEL DE INSTRUCTIUNE

Măsoară procentul de instrucțiuni care au fost executate cel puțin o dată în timpul unui set de teste.

TOOL STANDARD INTELLIJ



Current scope: all classes

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	86.7% (39/45)	83.1% (148/178)	84.1% (371/441)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
ro.boa.clinic	100% (1/1)	50% (1/2)	50% (1/2)
ro.boa.clinic.configuration	100% (1/1)	100% (9/9)	100% (27/27)
ro.boa.clinic.controller	100% (5/5)	82.4% (14/17)	86.2% (50/58)
ro.boa.clinic.dto	76.9% (10/13)	76.9% (30/39)	86.4% (57/66)
ro.boa.clinic.exception	57.1% (4/7)	57.1% (4/7)	57.1% (4/7)
ro.boa.clinic.model	100% (8/8)	83.3% (45/54)	87% (60/69)
ro.boa.clinic.runner	100% (2/2)	100% (6/6)	100% (39/39)
ro.boa.clinic.service	100% (8/8)	88.6% (39/44)	76.9% (133/173)

generated on 2024-04-15 13:25

Procentaje privind acoperirea la nivel de instrucțiune a claselor, metodelor și instrucțiunilor.
Informații sub forma de HTML-uri privind exact ce instrucțiuni sunt parcurse și ce instrucțiuni nu sunt accesate de către teste.

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/07.-Acoperire-la-nivel-de-instructiune

JACOCO vs TOOL INTELLIJ

INTELLIJ

```java
@Getter
@Getter
public final class PatientDetailsDto extends ProfileDetailsDto {
    @NonNull
    private String firstName;
    @NonNull
    private String lastName;
    @NonNull
    private String sex;
    @NonNull
    private LocalDate birthdate;

    public PatientDetailsDto(@NonNull Long id, @NotNull String firstName, @NotNull String lastName,
                             @NotNull String sex, @NotNull LocalDate birthdate) {
        super(id);
        this.firstName = firstName;
        this.lastName = lastName;
        this.sex = sex;
        this.birthdate = birthdate;
    }

    public static PatientDetailsDto fromPatient(@NonNull Patient patient) {
        return new PatientDetailsDto(
                patient.getId(),
                patient.getFirstName(),
                patient.getLastName(),
                patient.getSex().name(),
                patient.getBirthdate()
        );
    }
}
```

JACOCO

```java
@Getter
@Getter
public final class PatientDetailsDto extends ProfileDetailsDto {
    @NonNull
    private String firstName;
    @NonNull
    private String lastName;
    @NonNull
    private String sex;
    @NonNull
    private LocalDate birthdate;

    public PatientDetailsDto(@NonNull Long id, @NotNull String firstName, @NotNull String lastName,
                             @NotNull String sex, @NotNull LocalDate birthdate) {
        super(id);
        this.firstName = firstName;
        this.lastName = lastName;
        this.sex = sex;
        this.birthdate = birthdate;
    }

    public static PatientDetailsDto fromPatient(@NonNull Patient patient) {
        return new PatientDetailsDto(
                patient.getId(),
                patient.getFirstName(),
                patient.getLastName(),
                patient.getSex().name(),
                patient.getBirthdate()
        );
    }
}
```

IntelliJ IDEA are o performanță superioară față de JaCoCo la categoriile Class, Method și Line.

JaCoCo nu reușește să testeze deloc anumite părți ale codebase-ului

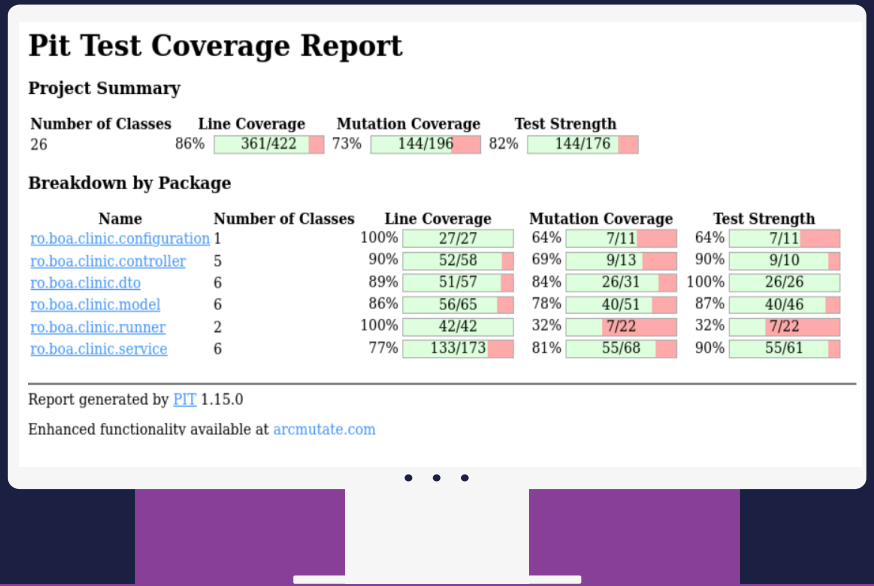
Atât IntelliJ IDEA cât și JaCoCo nu acoperă generarea automată a setter-ilor formați de linia de cod @Setter

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/07.-Acoperire-la-nivel-de-instructiune

MUTATION TESTING – PIT TEST

Mutational testing este o tehnică de testare a software-ului în care se introduc deliberat mici modificări (numite mutații) în codul sursă pentru a evalua calitatea setului de teste. Scopul este de a verifica dacă testele identifică aceste modificări, acesti mutanti.

Pentru a face mutation testing pe proiectul nostru in java am folosit plugin-ul **Pitest**. Raportul din imagine a fost generat pentru cele 5 clase de teste de integrare si clasa de teste unitare PatientServiceTest.



Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
26	86%	361/422	73%	144/196	82%	144/176

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
ro.boa.clinic.configuration	1	100%	27/27	64%	7/11	64%	7/11
ro.boa.clinic.controller	5	90%	52/58	69%	9/13	90%	9/10
ro.boa.clinic.dto	6	89%	51/57	84%	26/31	100%	26/26
ro.boa.clinic.model	6	86%	56/65	78%	40/51	87%	40/46
ro.boa.clinic.runner	2	100%	42/42	32%	7/22	32%	7/22
ro.boa.clinic.service	6	77%	133/173	81%	55/68	90%	55/61

Report generated by PIT 1.15.0

Enhanced functionality available at arcmutate.com

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/11.-Mutation-Testing

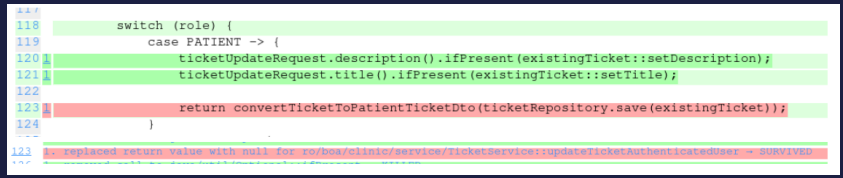
OMORAREA MUTANTILOR

```
String expectedJson = new JSONObject()
        .put("id", ticket.getId())
        .put("title", newTicketTitle)
        .put("description", newTicketDescription)
        .put("specialization", ticket.getSpecialization())
        .put("status", newTicketStatus)
        .put("response", null)
        .toString();

mockMvc.perform(requestTester.authenticatedPatch("/tickets/" + ticket.getId(), updateDto))
    .andExpect(status().isOk()).andExpect(content().json(expectedJson));

assertEquals(newTicketTitle, ticket.getTitle());
assertEquals(newTicketDescription, ticket.getDescription());
assertEquals(newTicketStatus, ticket.getStatus());
```

MUTANT



Pentru aceasta functie exista deja un test de integrare, dar el verifica doar ca ticketul si-a facut update in baza de date, nu verifica ce returneaza endpoint-ul. De aceea mutantul care face hardcode cu null a trait. Pentru a omorî mutantul vom face update la testul de integrare verificand si json-ul primit de la request.

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/11.-Mutation-Testing

COVERAGE vs MUTATION TESTING

MUTATION

```
30      public Account createPatientAccount(String email, String password) throws DataIntegrityViolationException {
31          return createAccount(email, password, Role.PATIENT);
32      }
33
34      /**
35       * @param email    the email address of the account
36       * @param password the plain text password
37       * @return the created account entity
38       * @throws DataIntegrityViolationException the email address is likely already used
39       */
40      public Account createAccount(String email, String password, Role role) throws DataIntegrityViolationException {
41          log.info("Creating a new account");
42          final String hashedPassword = passwordEncoder.encode(password);
43          Account newAccount = new Account(email, hashedPassword, role);
44
45          return accountRepository.save(newAccount);
46      }
47
48      public Optional<Account> findAccountByEmail(String email) {
49          return accountRepository.findByEmail(email);
50      }
51
52      public String getAuthenticatedUserEmail() {
53          log.info("Getting authenticated user account");
54          Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
55          if (!(authentication instanceof AnonymousAuthenticationToken)) {
56              return authentication.getName();
57          }
58          return null;
59      }
```

COVERAGE

```
30      public Account createPatientAccount(String email, String password) throws DataIntegrityViolationException {
31          return createAccount(email, password, Role.PATIENT);
32      }
33
34      /**
35       * @param email    the email address of the account
36       * @param password the plain text password
37       * @return the created account entity
38       * @throws DataIntegrityViolationException the email address is likely already used
39       */
40      public Account createAccount(String email, String password, Role role) throws DataIntegrityViolationException {
41          log.info("Creating a new account");
42          final String hashedPassword = passwordEncoder.encode(password);
43          Account newAccount = new Account(email, hashedPassword, role);
44
45          return accountRepository.save(newAccount);
46      }
47
48      public Optional<Account> findAccountByEmail(String email) {
49          return accountRepository.findByEmail(email);
50      }
51
52      public String getAuthenticatedUserEmail() {
53          log.info("Getting authenticated user account");
54          Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
55          if (!(authentication instanceof AnonymousAuthenticationToken)) {
56              return authentication.getName();
57          }
58          return null;
59      }
```

Procente de acoperire la nivel de instrucțiune aproximativ egale (diferenta maxima 4%)

Generatorul de muntanți a obținut rezultate mai bune sau egale decât testele de acoperire, în ciuda diferențelor mici

Acoperirea instrucțiunilor de return a fost diferită între cele două metode (exemplu imagine)

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/11.-Mutation-Testing

ACOPERIRE LA NIVEL DE CONDITIE

Metrică folosită în testarea software pentru a verifica cât de complet sunt evaluate condițiile dintr-un program în timpul testării. Scopul este verificarea fiecarei condiții individuale.

```
public Patient getAuthenticatedPatientProfile() {
    log.info("Getting authenticated patient profile");
    var userEmail = accountService.getAuthenticatedUserEmail();
    var patientProfile = patientRepository.findPatientProfileByEmail(userEmail);
    if (patientProfile == null) {
        throw new PatientProfileNotFoundException();
    } else {
        return patientProfile;
    }
}
```

Test 1: este creat profilul de pacient, se intră pe ramura condiției în care există profilul, caz în care acesta este returnat => acoperire 50% din necesar
Test 2: este oferit emailul unui cont care nu există, se intră pe ramura în care nu există profilul, caz în care este returnată eroarea => acoperire restul de 50%

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/09.-Acoperirea-la-nivel-de-condi%C8%9Bie

ACOPERIRE LA NIVEL DE CIRCUITE INDEPENDENTE

```
public TicketResponseDto updateTicketAuthenticatedUser(Long id, TicketUpdateRequestDto ticketUpdateRequest) {
        var role = accountService.getAuthenticatedUserAccount().getRole();

        switch (role) {
            case PATIENT -> {
                if (!isTicketOwnedByLoggedInPatient(existingTicket)) {
                    throw new TicketNotFoundException();
                }
            }
            case DOCTOR -> {
                if (!isTicketOwnedByLoggedInDoctor(existingTicket)) {
                    throw new TicketNotFoundException();
                }
            }
        }

        log.info("Updating the ticket");

        switch (role) {
            case PATIENT -> {
                ticketUpdateRequest.description().ifPresent(existingTicket::setDescription);
                return convertTicketToPatientTicketDto(ticketRepository.save(existingTicket));
            }
            case DOCTOR -> {
                ticketUpdateRequest.specialization().ifPresent(existingTicket::setSpecialization);
                return convertTicketToDoctorTicketDto(ticketRepository.save(existingTicket));
            }
            default -> throw new UnauthorizedAccessException();
        }
    }
```

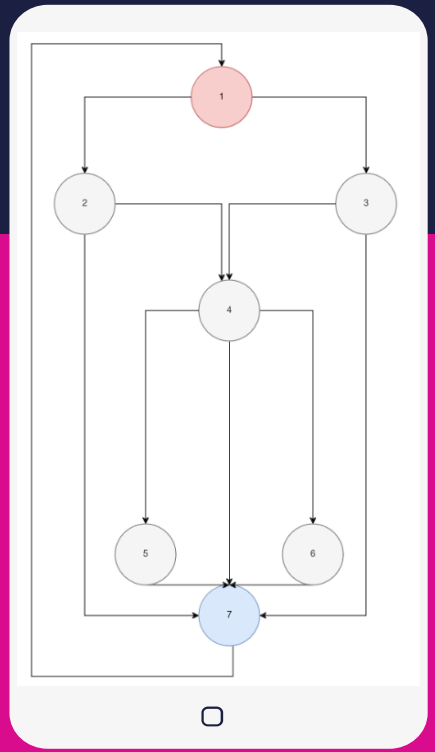
https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/10.-Acoperirea-la-nivel-de-circuite-independente

ACOPERIRE LA NIVEL DE CIRCUITE INDEPENDENTE

- măsurarea gradului în care instrucțiunile sau deciziile din codul sursă al unui program sunt executate în timpul testării
- identificare a limitei superioare pentru numărul de căi necesare obținerii unei acoperiri la nivel de ramură

Formula **McCabe** pentru Complexitate Ciclomatică -> numărul de circuite linear independente este intr-un graf complet conectat cu e arce si n noduri este **e-n+1**

Exemplu: n=7, e=12 => 6 circuite independente => 6 teste

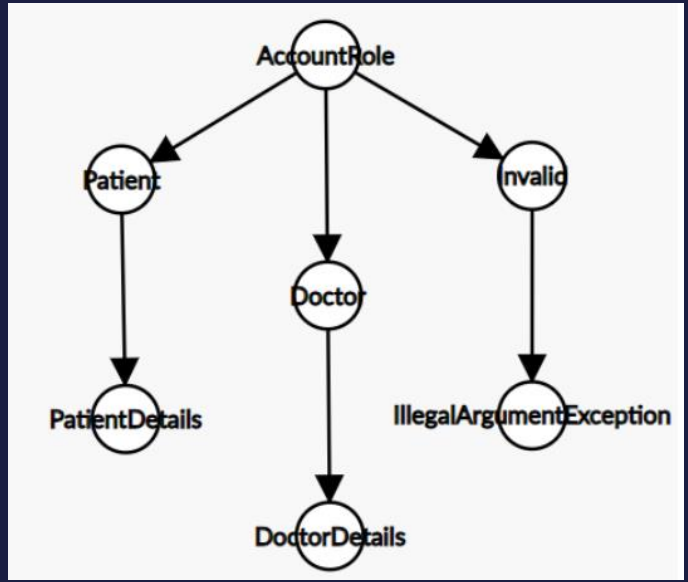


https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/10.-Acoperirea-la-nivel-de-circuite-independente

ACOPERIRE LA NIVEL DE DECIZIE

```java
@RequiredArgsConstructor  4 usages  Qmpzlawasd
@Service
public class ProfileService {
    private final PatientService patientService;
    private final DoctorService doctorService;

    public Optional<? extends ProfileDetailsDto> getProfileDetailsForAccount(@NonNull Account account) {  4 usages
        return switch (account.getRole()) {
            case PATIENT -> patientService.getPatientProfileForAccount(account).map(PatientDetailsDto::fromPatient);
            case DOCTOR -> doctorService.getDoctorProfileByAccount(account).map(DoctorDetailsDto::fromDoctor);
            default -> throw new IllegalArgumentException("Account must have a patient or a doctor profile");
        };
    }
}
```



Metodă de testare al cărei scop este să se asigure că fiecare ramură posibilă al fiecărui punct de decizie este executată cel puțin o dată și ca tot codul accesibil este executat.

switch ... case ⇔ if ... else ➲ am testat fiecare dintre aceste ramuri, indiferent de probabilitatea de acces pe ramură

https://github.com/irinaenescu2002/Testarea-Sistemelor-Software-2024/wiki/08.-Acoperire-la-nivel-de-decizie

REFERINTE

1. https://pitest.org/ (ultima accesare 11 mai 2024)
2. https://junit.org/junit5/ (ultima accesare 11. mai 2024)
3. https://www.eclemma.org/jacoco/ (ultima accesare 11 mai 2024)
4. Baeldung. (8 ianuarie 2024). ”A Guide to Spring Boot Testing”. https://www.baeldung.com/spring-boot-testing (ultima accesare 11 mai 2024)
5. Baeldung. (7 iulie 2023). ”Mutation Testing with PITest”. https://www.baeldung.com/java-mutation-testing-with-pitest (ultima accesare 11 mai 2024)
6. Geeks for Geeks. (11 martie 2022). “Spring Boot - Architecture” https://www.geeksforgeeks.org/spring-boot-architecture/ (ultima accesare 11 mai 2024)
7. https://openai.com/chatgpt/ (ultima accesare 11 mai 2024)

MULTUMIM!

GRUPA 333
Diaconu Stefan-Mircea, Dina George-Alexandru, Enescu Irina-Stefania,
Mihaila Nicolae, Potanga Alexandru-Alin