WHITE BOX TESTIRANJE

Klasa: IznajmljivacController.cs
Metoda:

```
// GET: Iznajmljivac/Edit/5
3 references | Please sign-in to New Relic CodeStream to see Code Level Metrics
public async Task<IActionResult> Edit(string id)
{
    if (id == null)
    {
        return NotFound();
    }

    var iznajmljivac = await _context.Iznajmljivaci.FindAsync(id);
    if (iznajmljivac == null)
    {
        return NotFound();
    }
    return View(iznajmljivac);
}
```

1) **Obuhvat iskaza/linija:**

TC1: (id = "1")

TC2: (id = "5")

TC3: (id = null)

Pokrivenost iskaza = ukupan broj iskaza obuhvaćen testovima/ukupan broj iskaza u programu

= 6 / 6 = 1 * 100% = 100%

U ovu svrhu napisani su testovi:

```
[TestMethod]
public async Task Edit_ValidId_ReturnsModel()
{
    var controller = new IznajmljivacController (_dbContext);
    var result = await controller.Edit("1") as ViewResult;
    Assert.IsNotNull(result);
    var model = result.Model as Iznajmljivac;
    Assert.AreEqual("1", model.Id);
}
[TestMethod]
public async Task Edit_InvalidId_ReturnsNotFound()
{
    var controller = new IznajmljivacController(_dbContext);
    var result1 = await controller.Edit("5") as NotFoundResult;
    Assert.IsNotNull(result1);
    Assert.AreEqual(404, result1.StatusCode);
}
[TestMethod]
public async Task Edit_InvalidIdNull_ReturnsNotFound()
{
    var controller = new IznajmljivacController(_dbContext);
    var result2 = await controller.Edit(null) as NotFoundResult;
    Assert.IsNotNull(result2);
    Assert.AreEqual(404, result2.StatusCode);
}
```

2) **Obuhvat odluka/grana**

   Gore navedeni skup testova postiže i 100% obuhvat odluka/grana.

3) **Obuhvat uslova**

   S obzirom da u if iskazima metode nema složenih uslova, postignut je i potpuni obuhvat uslova.

4) **Modifikovani uslov/odluka**

5) **Obuhvat petlji**

Linija `var iznajmljivac = await _context.Iznajmljivaci.FindAsync(id);` se može napisati kao:

```
Iznajmljivac iznajmljivac = null;

foreach (var item in _context.Iznajmljivaci)
{
    if (item.Id == id)
    {
        iznajmljivac = item;
        break;
    }
}
```

U TestInitialize metodi je dodano 5 objekata.

```
[TestInitialize]
public void Setup()
{
    var options = new DbContextOptionsBuilder<ApplicationDbContext>()
        .UseInMemoryDatabase(databaseName: Guid.NewGuid().ToString())
        .Options;
    _dbContext = new ApplicationDbContext(options);
    iznajmljivac = new Iznajmljivac { Id = "3", UserName = "Iznajmljivac3", Email =
"user3@example.com" };
    _dbContext.Iznajmljivaci.AddRange(
        new Iznajmljivac { Id = "1", UserName = "Iznajmljivac1", Email =
"user1@example.com" },
        new Iznajmljivac { Id = "2", UserName = "Iznajmljivac2", Email =
"user2@example.com" },
        iznajmljivac,
      new Iznajmljivac { Id = "4", UserName = "Iznajmljivac4", Email =
"user4@example.com" },
      new Iznajmljivac { Id = "6", UserName = "Iznajmljivac6", Email =
"user6@example.com" }
    );
    _dbContext.SaveChanges();
}
```

Možemo proširiti skup testova na:
TC1: (id = “1”)
TC2: (id = “5“)
TC3: (id = null)
TC4: (id = “2”)
TC5: (id = “3”)
TC6: (id = “4”)
TC7: (id = “6”)

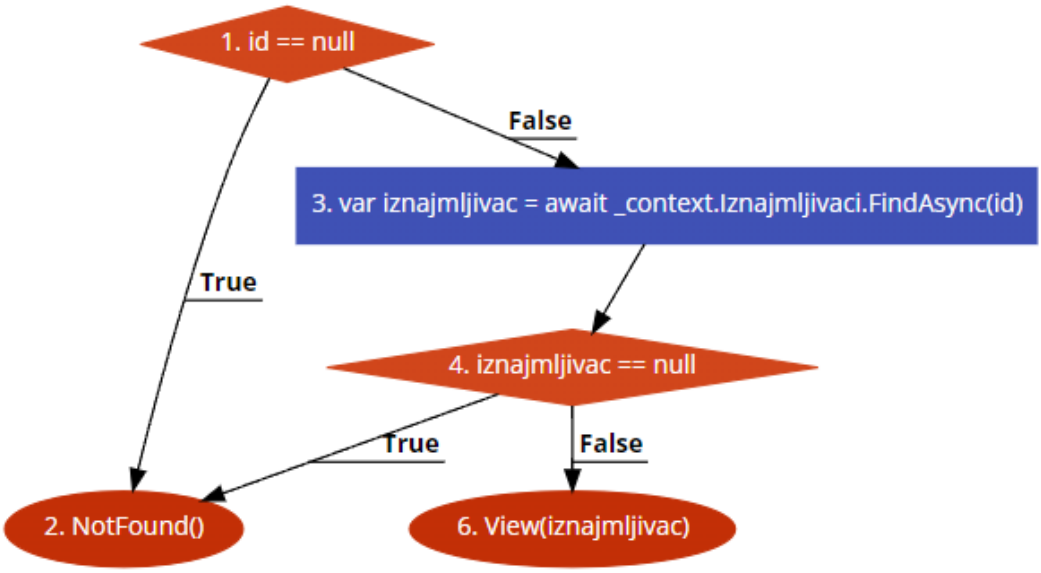
Preskočiti tijelo petlje:
TC3
Uradi 1 prolaz kroz petlju:
TC1
Uradi 2 prolaza kroz petlju:
TC4
Uradi slučajan broj prolaza kroz petlju:
TC5
Uradi n, n-1, n+1 prolaza kroz petlju:
TC7, TC6, TC2

6) Obuhvat toka podataka

Ova tehnika nije pogodna za izabranu metodu jer ona nema ugnježdenih if iskaza i petlji

7) Obuhvat puteva

Za obuhvat puteva je potreban dijagram toka:



Uočavamo puteve

1. 1-2
2. 1-3-4-2
3. 1-3-4-6

Potrebna su 3 testa za potpuni obuhvat linija, kao i za potpuni obuhvat puteva.

Za ovaj graf V(G) = E – N + 2p; E = 6 (ubrajamo i ivicu koja „ulazi“ u prvi ćvor), N = 5, p = 1

= 6-5+2 = 3

Ovo indicira da je maksimalan broj neovisnih puteva 3.

WHITE BOX TESTIRANJE

Klasa: RecenzijaController.cs

Metoda: public async Task<IActionResult> LeaveReview(string? id)

```
// Ostavljanje recenzija
[Authorize(Roles = "ObicniKorisnik")]
3 references | Please sign-in to New Relic CodeStream to see Code Level Metrics | 1/1 passing
public async Task<IActionResult> LeaveReview(string? id)
{
    if (id == null) return NotFound();
    var artist = await _context.Izvodjaci.FirstOrDefaultAsync(m => m.Id == id);
    if (artist == null) return NotFound();
    var listaRecenzija = await _context.Recenzije.Where(r => r.izvodjacId == id).ToListAsync();
    int brojac = 0;
    for(int i = 0; i < listaRecenzija.Count; i++)
    {
        if (listaRecenzija[i].izvodjacId == id) brojac++;
    }
    var recenzija = new Recenzija();
    recenzija.izvodjac = artist;
    recenzija.izvodjacId = artist.Id;
    recenzija.rating = 5;
    return View(recenzija);
}
```

Pristupi planiranja:

1) Obuhvat iskaza/linija:

   TC1: (id = “null”)
   TC2: (id = “${broj}”, pri čemu je broj takav da nema nijedan izvođač sa tim id-em)
   TC3: (id = “${broj}”, pri čemu je broj takav da postoji izvođač sa tim id-em)

   Pokrivenost iskaza = ukupan broj iskaza obuhvaćen testovima/ukupan broj iskaza u programu
   = 3 / 3 = 1 * 100% = 100%

2) Obuhvat odluka/grana

   TC1: (id = “null”)
   TC2: (id = “${broj}”, pri čemu je broj takav da nema nijedan izvođač sa tim id-em)
   TC3: (id = “${broj}”, pri čemu je broj takav da postoji izvođač sa tim id-em)

   Pokrivenost grana = 100%

3) Obuhvat uslova

Nema kompleksnih if uslova za analizu obuhvata uslova.

4) Modifikovani uslov/odluka obuhvat
   Nema kompleksnih if uslova za analizu modifikovanog uslov/odluka obuhvata.

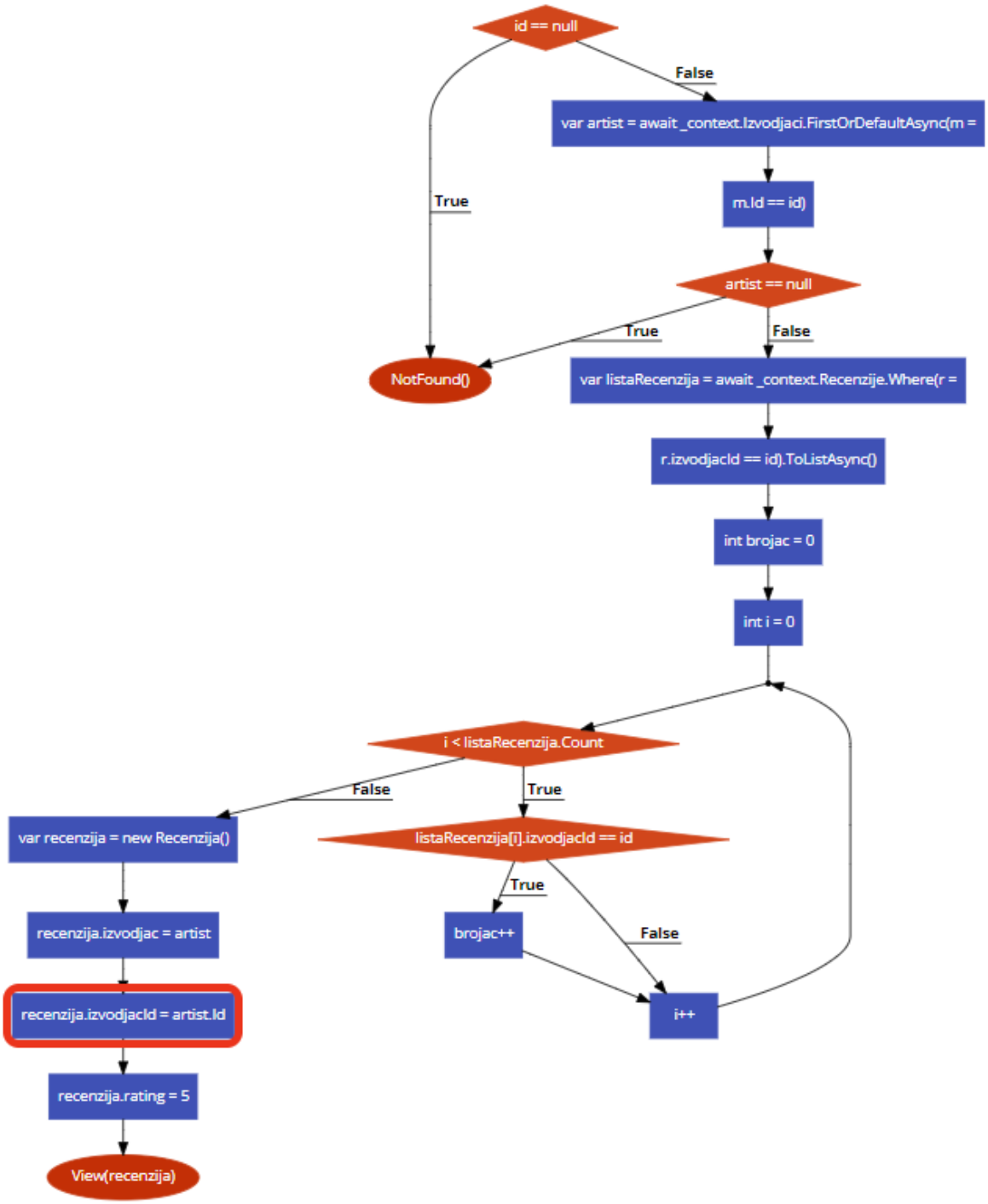
5) Obuhvat petlji

   TC1: (id = “${broj}”, pri čemu je broj takav da nema nijedna recenzija za izvođača sa tim id-em)
   TC2: (id = “${broj}”, pri čemu je broj takav da ime jedna recenzija za izvođača sa tim id-em)
   TC3: (id = “${broj}”, pri čemu je broj takav da ima dvije recenzije za izvođača sa tim id-em)
   TC4: (id = “${broj}”, pri čemu je broj takav da ima slučajan broj veći od dva recenzija za izvođača sa tim id-em)

   Preskočiti tijelo petlje
   TC1
   Uradi 1 prolaz kroz petlju
   TC2
   Uradi 2 prolaza kroz petlju
   TC3
   Uradi slučajan broj prolaza kroz petlju
   TC4
   Uradi n, n-1, n+1 prolaza kroz petlju
   TC2, TC1, TC3

6) Obuhvat toka podataka

7) Obuhvat puteva



8 različitih puteva, najmanje 8 testnih slučajeva
Puni linijski obuhvat se može postići sa minimalno 2 puta

Omjer test slučajeva koji se zahtijevaju za test sistema sa punim obuhvatom linija (3 testa) nasuprot punog obuhvata puteva (8 testova) je 3:8
$V(G) = E – N + 2p$, $E = 10$, $N = 9$, $p = 1$, $V(G) = 10 – 9 + 2 = 3$

WHITE BOX TESTIRANJE

Klasa: KoncertController.cs

Metoda:

```
public async Task<IActionResult> Edit(int id, [Bind("Id,naziv,izvodjacId,zanr")] Koncert koncert)
{
    if (id != koncert.Id)     1
    {
        return NotFound();    2
    }

    if (ModelState.IsValid)   3
    {
        try       4
        {
            _context.Update(koncert);   5
            await _context.SaveChangesAsync();   6
        }
        catch (DbUpdateConcurrencyException)    7
        {
            if(!KoncertExists(koncert.Id)) 8
            {
                return NotFound();    9
            }
        }
        return RedirectToAction(nameof(Index));    10
    }
    return View(koncert);   11
}
```

Pristupi planiranja:

1) Obuhvat iskaza/linija:

   TC1: (id= 1, new Koncert { id=1, naziv=”concert”, izvodjacId=1, zanr=”rok”})
   Ovim pozivom obuhvatamo linije 3, 4, 5, 6, i 10. Primjetimo da je ovo ispravan poziv gdje se uspješno vrši ažuriranje. Sve pod pretpostavkom da id postoji u bazi.

   TC2: (id= 7, new Koncert {id=2, naziv=”naziv”, izvodjacId=2, zanr=”folk”})
   Ovim pozivom obuhvatamo linije 1 i 2, to je slučaj kada nam nisu jednaki id iz parametra i id iz koncerta. Tada treba da se vrati NotFound.

   TC3: (id=16, new Koncert { id=16, naziv=”nešto”})
   Ovim pozivom obuhvatamo slučaj kada ModelState nije validan pa samim time onda pokrivamo samo liniju 11. ModelState nije validan jer nam fali podataka iz klase Koncert.

   TC4: (id=17, new Koncert{id=17, naziv=”neki”, izvodjacId=12, zanr=”rok”})
   Pod pretpostavkom da u bazi nema id sa vrijednošću 17, tada bi trebao da se baci izuzetak i da se uhvati u catch bloku, pa time pokrivamo preostale linije koda a to su 7,8 i 9.
   Primjetimo da smo ovime pokrili sve linije.

Pokrivenost iskaza = 4/4=1*100%=100%

2) Obuhvat odluka/grana:

   TC1: (id= 1, new Koncert { id=1, naziv=”concert”, izvodjacId=1, zanr=”rok”})
   Ovim pozivom obuhvatamo granu 3-4..6 . Ovo je ispravan poziv gdje se uspješno vrši ažuriranje. Sve pod pretpostavkom da id postoji u bazi.

   TC2: (id= 7, new Koncert {id=2, naziv=”naziv”, izvodjacId=2, zanr=”folk”})
   Ovim pozivom obuhvatamo granu 1..2, to je slučaj kada nam nisu jednaki id iz parametra i id iz koncerta. Tada treba da se vrati NotFound.

   TC3: (id=16, new Koncert {id=16, naziv=”nešto”})
   Ovim pozivom obuhvatamo granu koja se aktivira samo kada ModelState nije validan to je else grana koja sadrži samo liniju 11.

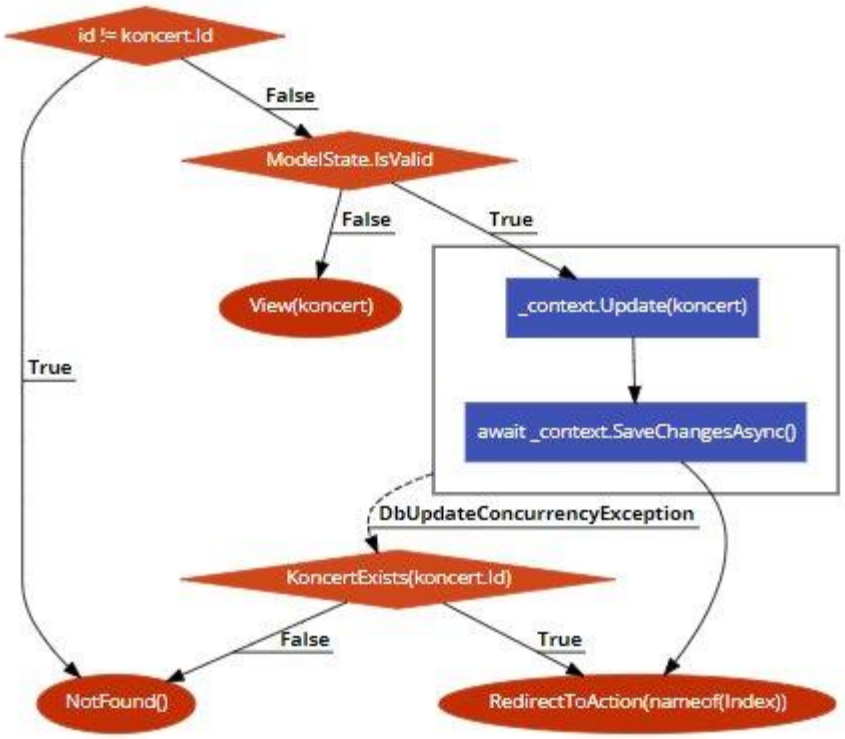
   TC4: (id=17, new Koncert{id=17, naziv=”neki”, izvodjacId=12, zanr=”rok”})
   Pod pretpostavkom da u bazi nema id sa vrijednošću 17, tada bi trebao da se baci izuzetak i da se uhvati u catch bloku, pa time pokrivamo preostalu granu 3-7..9. Ovime smo pokrili sve grane.

   Pokrivenost grana: 100%

3) Obuhvat uslova

4) Modifikovani uslov/odluka obuhvat:
5) Obuhvat petlji:
   Nema petlji.
6) Obuhvat toka podataka:

7) Obuhvat puteva:



5 različitih puteva, najmanje 5 testnih slučajeva.
Puni linijski obuhvat se može postići sa minimalno 4 puta
Omjer test slučajeva koji se zahtijevaju za test sistema sa punim obuhvatom linija (4 testa) nasuprot punog obuhvata puteva (5 testova) je 4:5=80%
V(G) = E – N + 2p, E = 9, N = 8, p = 1, V(G) = 9 – 8 + 2 = 3

WHITE BOX TESTIRANJE

Klasa: IzvodjacController.cs

Metoda:

```
[HttpPost]
[ValidateAntiForgeryToken]
6 references | Please sign-in to New Relic CodeStream to see Code Level Metrics
public async Task<IActionResult> Edit(string id, [Bind("Id,UserName,Email")] Izvodjac izvodjac)
{
    if (id != izvodjac.Id) 1
    {
        return NotFound(); 2
    }


    if (ModelState.IsValid) 3
    {
        try 4
        {
            _context.Update(izvodjac); 5
            await _context.SaveChangesAsync(); 6
        }
        catch (DbUpdateConcurrencyException) 7
        {
            if (!IzvodjacExists(izvodjac.Id)) 8
            {
                return NotFound(); 9
            }
        }
        return RedirectToAction(nameof(Index)); 10
    }
    return View(izvodjac); 11
}
```

Pristupi planiranja:

1) Obuhvat iskaza/linija:

   TC1: (id= 1, new Izvodjac { Id=1, UserName=”izvodjac1”, Email=izvodjac1@gmail.com })
   Ovim pozivom obuhvatamo linije 1, 3, 4, 5, 6, i 10. Ovo je ispravan poziv, dakle uspješno se vrši ažuriranje. Ovo vrijedi pod pretpostavkom da izvodjac sa id-em 1 postoji u bazi.

   TC2: (id= 2, new Izvodjac { Id=1, UserName=”izvodjac1”, Email=izvodjac1@gmail.com })
   Ovim pozivom obuhvatamo linije 1 I 2, to je slučaj kada nisu jednaki id iz parametra i id izvodjaca. Tada se vraća NotFound.

   TC3: (id=1, new Izvodjac { Id=1})
   Ovim pozivom obuhvatamo slučaj kada ModelState nije validan pa samim time onda pokrivamo linije 1, 3 i 11.

TC4: (id=5, new Izvodjac { Id=5, UserName=”izvodjac1”, Email=izvodjac1@gmail.com })
Pod pretpostavkom da u bazi nema Izvodjac sa id-em 5, tada bi trebalo da se baci DbUpdateConcurrencyException, te da se uhvati u catch bloku. Linije koda pokrivene ovim testnim slučajem su 1,3,4,5,7,8,9

Ovim smo pokrili sve linije
Pokrivenost iskaza = 4/4=1*100%=100%

2) Obuhvat odluka/grana

TC1: (id= 1, new Izvodjac { Id=1, UserName=”izvodjac1”, Email=izvodjac1@gmail.com })
Ovaj testni slučaj obuhvata granu 3-4..6 . Ovo je ispravan poziv gdje se uspješno vrši ažuriranje. Ovo radi pod pretpostavkom da Izvodjac sa id-em 1 postoji u bazi.

TC2: (id= 2, new Izvodjac { Id=1, UserName=”izvodjac1”, Email=izvodjac1@gmail.com })
Ovim slučajem obuhvatamo granu 1..2. Id iz parametra i id Izvodjaca nisu jednaki pa se tada vraća NotFound.

TC3: (id=1, new Izvodjac { Id=1})
Ovaj slučaj pokriva granu koja pokriva slučaj za neispravan ModelState. Ovim je pokrivena grana koja sadrži samo liniju 11.

TC4: (id=5, new Izvodjac { Id=5, UserName=”izvodjac1”, Email=izvodjac1@gmail.com })
Pod pretpostavkom da u bazi nema Izvodjac sa id-em 5, tada bi trebalo da se baci DbUpdateConcurrencyException, te da se uhvati u catch bloku. Ovim je pokrivena grana 3-7..9

Pokrivenost grana: 100%

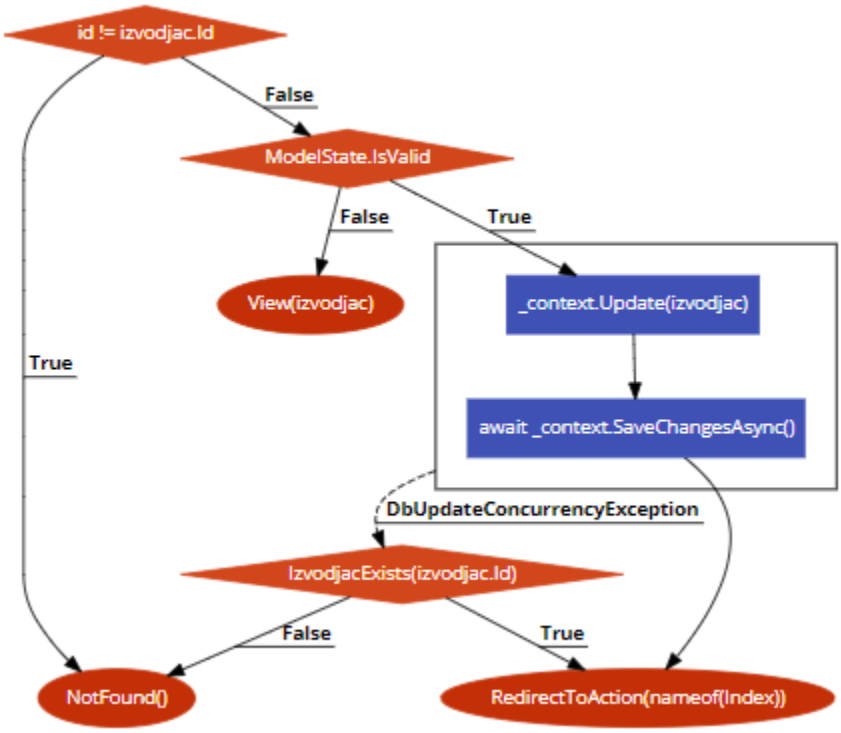
3) Obuhvat uslova

4) Modifikovani uslov/odluka obuhvat

5) Obuhvat petlji
Nema petlji.

6) Obuhvat toka podataka

7) Obuhvat puteva



Postoji 5 različitih puteva, što rezultira sa najmanje 5 testnih slučajeva.
Puni linijski obuhvat može se postići sa minimalno 4 puta.
Omjer testnih slučajeva koji se zahtijevaju za test sistema sa punim obuhvatom linija (4 testa) nasuprot punog obuhvata puteva (5 testova) je 4:5
V(G) = E – N + 2p, E = 9, N = 8, p = 1, V(G) = 9 – 8 + 2 = 3

WHITE BOX TESTIRANJE

Klasa: KoncertManager.cs

Metoda: SortAktuelni(string? aktuelniSortOrder, string? searchString)

```
public async Task<IEnumerable<Koncert>> SortAktuelni(string? aktuelniSortOrder, string? searchString)
{
    var koncerti = await GetAll();
    if (!String.IsNullOrEmpty(searchString))
    {
        koncerti = koncerti.Where(s => s.naziv?.Contains(searchString) == true).ToList();
    }

    switch (aktuelniSortOrder)
    {
        case "name_desc":
            koncerti = koncerti.OrderByDescending(s => s.naziv).ToList();
            break;
        case "Date":
            koncerti = koncerti.OrderBy(s => s.datum).ToList();
            break;
        case "date_desc":
            koncerti = koncerti.OrderByDescending(s => s.datum).ToList();
            break;
        default:
            koncerti = koncerti.OrderBy(s => s.naziv).ToList();
            break;
    }

    return koncerti;
}
```

Pristupi planiranja:

1) Obuhvat iskaza/linija:

   TC1: (aktuelniSortOrder = “name_desc”, searchString = “Koncert”)
   TC2: (aktuelniSortOrder = “Date”, searchString = “Koncert”)
   TC3: (aktuelniSortOrder = “date_desc”, searchString = “Koncert”)
   TC4: (aktuelniSortOrder = “default”, searchString = “Koncert”)
   TC5: (aktuelniSortOrder = “name_desc”, searchString = null
   TC6: (aktuelniSortOrder = “Date”, searchString = null)
   TC7: (aktuelniSortOrder = “date_desc”, searchString = null)
   TC8: (aktuelniSortOrder = “default”, searchString = null)

   Pokrivenost iskaza = ukupan broj iskaza obuhvaćen testovima/ukupan broj iskaza u programu
   = 8 / 8 = 1 * 100% = 100%

2) Obuhvat odluka/grana

TC1: (aktuelniSortOrder = “name_desc”, searchString = “Koncert”)
TC2: (aktuelniSortOrder = “Date”, searchString = “Koncert”)
TC3: (aktuelniSortOrder = “date_desc”, searchString = “Koncert”)
TC4: (aktuelniSortOrder = “default”, searchString = “Koncert”)
TC5: (aktuelniSortOrder = “name_desc”, searchString = null
TC6: (aktuelniSortOrder = “Date”, searchString = null)
TC7: (aktuelniSortOrder = “date_desc”, searchString = null)
TC8: (aktuelniSortOrder = “default”, searchString = null)

Pokrivenost iskaza = ukupan broj iskaza obuhvaćen testovima/ukupan broj iskaza u programu
= 8 / 8 = 1 * 100% = 100%

3) Obuhvat uslova
Ovaj pristup planiranja nije moguće uraditi u navedenoj metodi zbog toga što ona ne sadrži niti jedan složeni logički uslov.

4) Modifikovani uslov/odluka obuhvat
Ovaj pristup planiranja nije moguće uraditi u navedenoj metodi zbog toga što ona ne sadrži niti jedan složeni logički uslov.

5) Obuhvat petlji

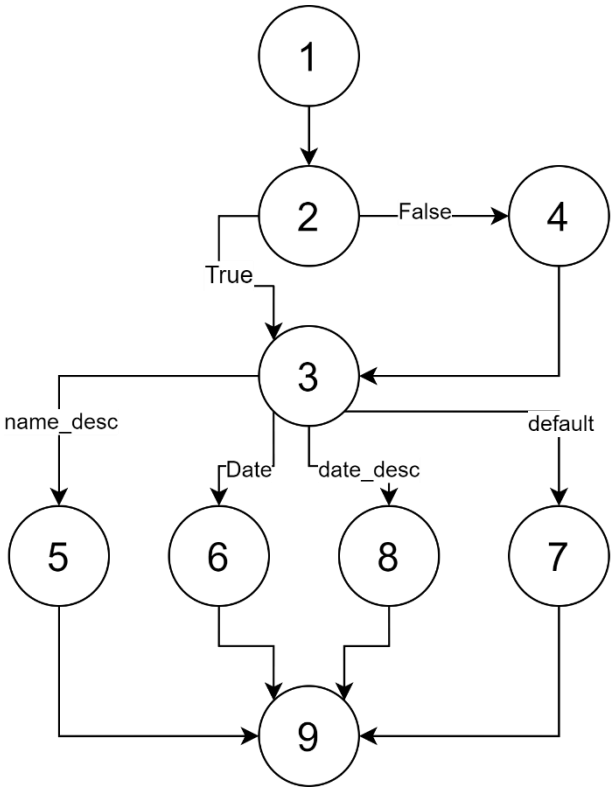
TC1: (aktuelniSortOrder = “name_desc”, searchString = null)
TC2: (aktuelniSortOrder = “name_desc”, searchString = “Koncert1”)
TC3: (aktuelniSortOrder = “name_desc”, searchString = “Koncert2”)
TC4: (aktuelniSortOrder = “name_desc”, searchString = “Nepostojeći koncert”)

Preskočiti tijelo petlje
TC1
Uradi 1 prolaz kroz petlju
TC2
Uradi 2 prolaza kroz petlju
TC3
Uradi slučajan broj prolaza kroz petlju
TC4
Uradi n, n-1, n+1 prolaza kroz petlju
TC2, TC3

6) Obuhvat toka podataka

Za dizajn testnih slučajeva biraju se putevi u skladu sa lokacijama definicije i korištenja varijabli. Ova tehnika nije praktična za intenzivno korištenje. Pogodna je za module sa ugniježdenim if iskazima i petljama.

7) Obuhvat puteva



8 različitih puteva, najmanje 8 testnih slučajeva
Puni linijski obuhvat se može postići sa minimalno 4 puta
Omjer test slučajeva koji se zahtijevaju za test sistema sa punim obuhvatom linija (4 testa) nasuprot punog obuhvata puteva (8 testova) je 1:2
V(G) = E – N + 2p, E = 12, N = 9, p = 1, V(G) = 12 – 9 + 2 = 5