1. ArticleServiceTest

Primjer aktivacije testnih repozitorija:

```
@InjectMocks 16 usages
private ArticleService articleService;

@Mock 10 usages
private FootwearService footwearService;

@Mock 10 usages
private ClothesService clothesService;

@Mock 9 usages
private LocatedatService locatedatService;

@Mock 32 usages
private ArticleRepository articleRepository;

@Mock 2 usages
private UserRepository userRepository;
```

Primjer pripreme prije svakog testa:

```
@BeforeEach bajan-bato
void setUp() {
    // Postavljanje artikala
    article1 = new Article();
    article1.setId(1);
    article1.setCategory("Boots");
    article1.setArticlename("Timberland");

    article2 = new Article();
    article2.setId(2);
    article2.setCategory("Sandals");
    article2.setArticlename("Birkenstock");

    // Postavljanje footwear objekata
    footwear1 = new Footwear();
    footwear1.setId(1);
    footwear1.setOpenness(null); // Prije dodavanja

    footwear2 = new Footwear();
    footwear2.setId(2);
    footwear2.setOpenness(null); // Prije dodavanja
}
```

Primjer testa:

```java
@Test  bajan-bato
void testAdd_Article() {
    // Arrange
    Article article = new Article();
    article.setId(2);
    when(articleRepository.save(article)).thenReturn(article);

    // Act
    Article result = articleService.add(article);

    // Assert
    assertNotNull(result);
    assertEquals( expected: 2, result.getId());
    verify(articleRepository, times( wantedNumberOfInvocations: 1)).save(article);
}
```

1.1 `testGet`

- **Opis testa**: Provjerava da li metoda `get` u `ArticleService` ispravno dohvaća `Article` objekt na temelju ID-a.
- **Ulazne varijable**: `id = 1`
- **Očekivani rezultat**: Metoda `get(1)` treba vratiti `Article` objekt s ID-jem 1.
- **Stvarni rezultat**: `Article` objekt nije `null`, ID je 1, metoda `findById` repository-ja pozvana je jednom.

1.2 `testAdd_Article`

- **Opis testa**: Provjerava da li metoda `add` u `ArticleService` ispravno dodaje novi `Article` objekt.
- **Ulazne varijable**: `Article` objekt s ID-jem 2
- **Očekivani rezultat**: Metoda `add(article)` treba spremiti `Article` objekt i vratiti isti objekt s ID-jem 2.
- **Stvarni rezultat**: `Article` objekt nije `null`, ID je 2, metoda `save` repository-ja pozvana je jednom.

1.3 `testRemove`

- **Opis testa**: Provjerava da li metoda `remove` u `ArticleService` ispravno uklanja `Article` objekt kada nema povezanih entiteta.

- **Ulazne varijable**: `articleId = 3`
- **Očekivani rezultat**: Metoda `remove(3)` treba pokušati ukloniti `Article` objekt s ID-jem 3 i vratiti `false` jer nema povezanih entiteta.
- **Stvarni rezultat**: Metoda vraća `false`, metode za dohvaćanje povezanih entiteta nisu pozvane, metoda `deleteById` nije pozvana.

1.4 `testAdd_WithParameters_Footwear`

- **Opis testa**: Testira dodavanje `Article` objekta tipa "footwear" koristeći metodu s parametrima u `ArticleService`.
- **Ulazne varijable**:
  - `nazivArtikla = "Patike"`
  - `slikaArtikla = "patike.jpg"`
  - `opcaKategorija = "Obuća"`
  - `kategorijaGoddoba = "Ljeto"`
  - `kategorijaLezernosti = "Neformalno"`
  - `glavnaBoja = "Crna"`
  - `sporednaBoja = "Bijela"`
  - `stanjeArtikla = "Novo"`
  - `sifKorisnika = 1`
  - `type = "footwear"`
- **Očekivani rezultat**: Metoda add treba dodati novi `Article` objekt tipa "footwear", postaviti sve parametre ispravno, vratiti `Article` objekt s ID-jem 4 i pozvati `footwearService.add` jednom.
- **Stvarni rezultat**: `Article` objekt nije `null`, ID je 4, svi parametri su ispravno postavljeni, metoda add `footwearService` pozvana je jednom, `clothesService` nije pozvan.

1.5 `testAdd_WithParameters_Clothes`

- **Opis testa**: Testira dodavanje `Article` objekta tipa "clothes" koristeći metodu s parametrima u `ArticleService`.
- **Ulazne varijable**:
  - `nazivArtikla = "Majica"`
  - `slikaArtikla = "majica.jpg"`
  - `opcaKategorija = "Odjeća"`
  - `kategorijaGoddoba = "Proljeće"`
  - `kategorijaLezernosti = "Formalno"`
  - `glavnaBoja = "Plava"`
  - `sporednaBoja = "Siva"`
  - `stanjeArtikla = "Polovno"`

    - `sifKorisnika = 2`
    - `type = "clothes"`
- **Očekivani rezultat**: Metoda add treba dodati novi `Article` objekt tipa "clothes", postaviti sve parametre ispravno, vratiti `Article` objekt s ID-jem 5 i pozvati `clothesService.add` jednom.
- **Stvarni rezultat**: `Article` objekt nije `null`, ID je 5, svi parametri su ispravno postavljeni, metoda add `clothesService` pozvana je jednom, `footwearService` nije pozvan.

1.6 `testAdd_ArticleWithType_Footwear`

- **Opis testa**: Provjerava dodavanje postojećeg `Article` objekta tipa "footwear" koristeći drugu varijantu add metode u `ArticleService`.
- **Ulazne varijable**:
    - `Article` objekt s ID-jem 6
    - `type = "footwear"`
- **Očekivani rezultat**: Metoda add treba dodati `Article` objekt tipa "footwear", vratiti isti objekt s ID-jem 6 i pozvati `footwearService.add` jednom.
- **Stvarni rezultat**: `Article` objekt nije `null`, ID je 6, metoda add `footwearService` pozvana je jednom, `clothesService` nije pozvan.

1.7 `testAdd_ArticleWithType_Clothes`

- **Opis testa**: Provjerava dodavanje postojećeg `Article` objekta tipa "clothes" koristeći drugu varijantu add metode u `ArticleService`.
- **Ulazne varijable**:
    - `Article` objekt s ID-jem 7
    - `type = "clothes"`
- **Očekivani rezultat**: Metoda add treba dodati `Article` objekt tipa "clothes", vratiti isti objekt s ID-jem 7 i pozvati `clothesService.add` jednom.
- **Stvarni rezultat**: `Article` objekt nije `null`, ID je 7, metoda add `clothesService` pozvana je jednom, `footwearService` nije pozvan.

1.8 `testGetAll`

- **Opis testa**: Provjerava da li metoda `getAll` u `ArticleService` ispravno vraća sve `Article` objekte.
- **Ulazne varijable**: Nema
- **Očekivani rezultat**: Metoda `getAll()` treba vratiti listu sa 3 `Article` objekta.
- **Stvarni rezultat**: Lista sa 3 `Article` objekta nije `null`, metoda `findAll` repository-ja pozvana je jednom.

1.9 `testGetByUserID`

- **Opis testa**: Provjerava da li metoda `getByUserID` u `ArticleService` ispravno dohvaća sve `Article` objekte povezane s određenim korisnikom.
- **Ulazne varijable**: `userId = 10`
- **Očekivani rezultat**: Metoda `getByUserID(10)` treba vratiti listu sa 2 `Article` objekta povezana s korisnikom ID 10.
- **Stvarni rezultat**: Lista sa 2 `Article` objekta nije `null`, metoda `getByUserID` repository-ja pozvana je jednom.

1.10 `testGetFeatured`

- **Opis testa**: Provjerava da li metoda `getFeatured` u `ArticleService` ispravno vraća mapu od 8 istaknutih `Article` objekata sa pripadajućim korisničkim podacima.
- **Ulazne varijable**: Nema
- **Očekivani rezultat**: Metoda `getFeatured()` treba vratiti mapu sa 8 `Article` objekata, svaki mapiran na odgovarajuće korisničke podatke.
- **Stvarni rezultat**: Metoda nije dovršena u trenutnom odgovoru, ali se očekuje da će rezultat biti `Map<Article, String>` s 8 unosa, gdje svaki ključ (`Article`) i vrijednost (`String`) nisu `null`, i metoda `userRepository.findById` pozvana je jednom za svaki članak.

1.11 `testGet_NonExistingArticle`

- **Opis testa**: Provjerava da li metoda `get` u `ArticleService` ispravno rukuje slučajem kada `Article` objekt s danim ID-jem ne postoji.
- **Ulazne varijable**: `id = 999`
- **Očekivani rezultat**: Metoda `get(999)` treba vratiti `null` jer `Article` s ID-jem 999 ne postoji.
- **Stvarni rezultat**: Metoda vraća `null`, metoda `findById` repository-ja pozvana je jednom.

1.12 `testAdd_InvalidType`

- **Opis testa**: Provjerava da li metoda `add` u `ArticleService` ispravno rukuje slučajem kada je tip artikla nevalidan.
- **Ulazne varijable**:
  - `Article` objekt s ID-jem 8
  - `type = "invalid_type"`

- **Očekivani rezultat**: Metoda `add(article, "invalid_type")` treba dodati `Article` objekt bez povezivanja s bilo kojim servisom (neće biti `Footwear` niti `Clothes`).
- **Stvarni rezultat**: `Article` objekt nije `null`, ID je 8, metoda `save` repository-ja pozvana je jednom, `footwearService.add` i `clothesService.add` nisu pozvani.

1.13 `testRemove_WithLocatedatEntries`

- **Opis testa**: Provjerava da li metoda `remove` u `ArticleService` ispravno rukuje uklanjanjem `Article` objekta kada postoje povezani `Locatedat` entiteti.
- **Ulazne varijable**: `articleId = 4`
- **Očekivani rezultat**: Metoda `remove(4)` treba ukloniti sve povezane `Locatedat` entitete i potom ukloniti `Article` objekt, vraćajući `false` jer `Article` nije direktno uklonjen.
- **Stvarni rezultat**: Metoda vraća `false`, međutim, metode za dohvaćanje i brisanje `Locatedat` entiteta nisu pozvane, metoda `deleteById` nije pozvana. (Primijetno da se u testu zapravo ne pozivaju ove metode zbog `assertFalse`)

1.14 `testAdd_WithNullType`

- **Opis testa**: Provjerava da li metoda `add` u `ArticleService` ispravno rukuje slučajem kada je tip artikla `null`.
- **Ulazne varijable**:
  - `Article` objekt s ID-jem 9
  - `type = null`
- **Očekivani rezultat**: Metoda `add(article, null)` treba dodati `Article` objekt bez povezivanja s bilo kojim servisom.
- **Stvarni rezultat**: `Article` objekt nije `null`, ID je 9, metoda `save` repository-ja pozvana je jednom, `footwearService.add` i `clothesService.add` nisu pozvani.

1.15 `testAdd_WithEmptyType`

- **Opis testa**: Provjerava da li metoda `add` u `ArticleService` ispravno rukuje slučajem kada je tip artikla prazan string.
- **Ulazne varijable**:
  - `Article` objekt s ID-jem 10
  - `type = ""`
- **Očekivani rezultat**: Metoda `add(article, "")` treba dodati `Article` objekt bez povezivanja s bilo kojim servisom.

- **Stvarni rezultat**: `Article` objekt nije `null`, ID je 10, metoda `save` repository-ja pozvana je jednom, `footwearService.add` i `clothesService.add` nisu pozvani.

1.16 `testAdd_WithFeaturedArticlesLessThan8`

- **Opis testa**: Provjerava da li metoda `getFeatured` u `ArticleService` ispravno rukuje slučajem kada je broj dostupnih istaknutih `Article` objekata manji od 8.
- **Ulazne varijable**: Nema
- **Očekivani rezultat**: Metoda `getFeatured()` treba vratiti mapu sa svim dostupnim istaknutim `Article` objektima (u ovom slučaju 3) umjesto da zahtijeva minimalno 8.
- **Stvarni rezultat**: `Map<Article, String>` nije `null`, veličina mape je 3, metoda `findAll` repository-ja pozvana je jednom.

2. ClothesServiceTest

2.1 `testGetById_WhenClothesExists`

- **Opis testa**: Provjerava da li metoda `getById` u `ClothesService` ispravno dohvaća `Clothes` objekt kada postoji u repository-ju.
- **Ulazne varijable**: `id = 1`

- **Očekivani rezultat**: Metoda `getById(1)` treba vratiti `Clothes` objekt s ID-jem 1.
- **Stvarni rezultat**: `Clothes` objekt nije `null`, ID je 1, metoda `findById` repository-ja pozvana je jednom.

2.2 `testGetById_WhenClothesDoesNotExist`

- **Opis testa**: Provjerava da li metoda `getById` u `ClothesService` ispravno rukuje slučajem kada `Clothes` objekt s danim ID-jem ne postoji.
- **Ulazne varijable**: `id = 3`
- **Očekivani rezultat**: Metoda `getById(3)` treba vratiti `null` jer `Clothes` s ID-jem 3 ne postoji.
- **Stvarni rezultat**: Metoda vraća `null`, metoda `findById` repository-ja pozvana je jednom.

2.3 `testGetAll`

- **Opis testa**: Provjerava da li metoda `getAll` u `ClothesService` ispravno vraća sve `Clothes` objekte.
- **Ulazne varijable**: Nema
- **Očekivani rezultat**: Metoda `getAll()` treba vratiti listu sa 2 `Clothes` objekta.
- **Stvarni rezultat**: Lista sa 2 `Clothes` objekta nije `null`, metoda `findAll` repository-ja pozvana je jednom.

2.4 `testAdd`

- **Opis testa**: Provjerava da li metoda `add` u `ClothesService` ispravno dodaje novi `Clothes` objekt.
- **Ulazne varijable**: `Clothes` objekt s ID-jem 3
- **Očekivani rezultat**: Metoda `add(newClothes)` treba spremiti `Clothes` objekt i vratiti isti objekt s ID-jem 3.
- **Stvarni rezultat**: `Clothes` objekt nije `null`, ID je 3, metoda `save` repository-ja pozvana je jednom.

2.5 `testUpdate`

- **Opis testa**: Provjerava da li metoda `update` u `ClothesService` ispravno ažurira postojeći `Clothes` objekt.
- **Ulazne varijable**: `Clothes` objekt s ID-jem 1 (ažurirani podaci)
- **Očekivani rezultat**: Metoda `update(updatedClothes)` treba ažurirati `Clothes` objekt i vratiti isti objekt s ID-jem 1.

- **Stvarni rezultat**: `Clothes` objekt nije `null`, ID je 1, metoda `save` repository-ja pozvana je jednom.

2.6 `testDelete_WhenClothesExists`

- **Opis testa**: Provjerava da li metoda `delete` u `ClothesService` ispravno uklanja `Clothes` objekt kada postoji u repository-ju.
- **Ulazne varijable**: `idToDelete = 1`
- **Očekivani rezultat**: Metoda `delete(1)` treba ukloniti `Clothes` objekt s ID-jem 1 i vratiti `true`.
- **Stvarni rezultat**: Metoda vraća `true`, metoda `existsById` repository-ja pozvana je jednom, metoda `deleteById` repository-ja pozvana je jednom, metoda `articleService.remove` pozvana je jednom.

2.7 `testDelete_WhenClothesDoesNotExist`

- **Opis testa**: Provjerava da li metoda `delete` u `ClothesService` ispravno rukuje slučajem kada `Clothes` objekt s danim ID-jem ne postoji.
- **Ulazne varijable**: `idToDelete = 3`
- **Očekivani rezultat**: Metoda `delete(3)` treba vratiti `false` jer `Clothes` s ID-jem 3 ne postoji.
- **Stvarni rezultat**: Metoda vraća `false`, metoda `existsById` repository-ja pozvana je jednom, metode `deleteById` i `articleService.remove` nisu pozvane.

2.8 `testAdd_WhenRepositoryThrowsException`

- **Opis testa**: Provjerava da li metoda `add` u `ClothesService` ispravno rukuje izuzetkom koji se baca pri spremanju `Clothes` objekta.
- **Ulazne varijable**: `Clothes` objekt s ID-jem 3
- **Očekivani rezultat**: Metoda `add(newClothes)` treba baciti `RuntimeException` s porukom "Database error".
- **Stvarni rezultat**: Izuzetak `RuntimeException` je bačen s porukom "Database error", metoda `save` repository-ja pozvana je jednom.

2.9 `testDelete_WhenArticleServiceThrowsException`

- **Opis testa**: Provjerava da li metoda `delete` u `ClothesService` ispravno rukuje izuzetkom koji se baca pri uklanjanju povezane `Article` entitete.
- **Ulazne varijable**: `idToDelete = 1`

- **Očekivani rezultat**: Metoda `delete(1)` treba baciti `RuntimeException` s porukom "Removal error".
- **Stvarni rezultat**: Izuzetak `RuntimeException` je bačen s porukom "Removal error", metode `existsById` i `deleteById` repository-ja pozvane su jednom, metoda `articleService.remove` pozvana je jednom.

3. FootwearServiceTest

3.1 `testAddFootwear_OpenCoverage`

- **Opis testa**: Testira dodavanje `Footwear` objekta s pokrivenošću "Open" koristeći `Scrapper` klasu u `FootwearService`.
- **Ulazne varijable**:
  - `Article` objekt s ID-jem 1, kategorija "Boots", naziv "Timberland"
  - `Footwear` objekt s ID-jem 1, `openness = null`
- **Očekivani rezultat**: Metoda `add(newFootwear)` treba postaviti `openness` na "Open", spremiti `Footwear` objekt i vratiti isti objekt s postavljenom vrijednošću.
- **Stvarni rezultat**: `Footwear` objekt nije `null`, `openness` je postavljen na "Open", metode `articleService.get(1)` i `footwearRepository.save` pozvane su jednom, `Scrapper.getCoverage("Boots", ...)` pozvana je jednom.

3.2 `testAddFootwear_ClosedCoverage`

- **Opis testa**: Testira dodavanje `Footwear` objekta s pokrivenošću "Closed" koristeći `Scrapper` klasu u `FootwearService`.
- **Ulazne varijable**:
  - `Article` objekt s ID-jem 2, kategorija "Sandals", naziv "Birkenstock"
  - `Footwear` objekt s ID-jem 2, `openness = null`
- **Očekivani rezultat**: Metoda `add(newFootwear)` treba postaviti `openness` na "Closed", spremiti `Footwear` objekt i vratiti isti objekt s postavljenom vrijednošću.
- **Stvarni rezultat**: `Footwear` objekt nije `null`, `openness` je postavljen na "Closed", metode `articleService.get(2)` i `footwearRepository.save` pozvane su jednom, `Scrapper.getCoverage("Sandals", ...)` pozvana je jednom.

3.3 `testAddFootwear_RainCoverage`

- **Opis testa**: Provjerava da li metoda `add` u `FootwearService` ispravno rukuje slučajem kada `Article` objekt ne postoji.
- **Ulazne varijable**:
  - `Article` objekt s ID-jem 3 nije pronađen (`null`)
  - `Footwear` objekt s ID-jem 3, `openness = null`
- **Očekivani rezultat**: Metoda `add(newFootwear)` treba baciti `NullPointerException` jer `Article` objekt nije pronađen.
- **Stvarni rezultat**: Metoda baca `NullPointerException`, metode `articleService.get(3)` pozvana je jednom, metode `footwearRepository.save` i `Scrapper` nisu pozvane.

3.4 `testAddFootwear_SnowCoverage`

- **Opis testa**: Testira dodavanje `Footwear` objekta s pokrivenošću "Snow" koristeći `Scrapper` klasu u `FootwearService`.
- **Ulazne varijable**:
  - `Article` objekt s ID-jem 4, kategorija "Slippers", naziv "Crocs"
  - `Footwear` objekt s ID-jem 4, `openness = null`
- **Očekivani rezultat**: Metoda `add(newFootwear)` treba postaviti `openness` na "Snow", spremiti `Footwear` objekt i vratiti isti objekt s postavljenom vrijednošću.
- **Stvarni rezultat**: `Footwear` objekt nije `null`, `openness` je postavljen na "Snow", metode `articleService.get(4)` i `footwearRepository.save` pozvane su jednom, `Scrapper.getCoverage("Slippers", ...)` pozvana je jednom.

3.5 `testGetById_WhenFootwearExists`

- **Opis testa**: Provjerava da li metoda `getById` u `FootwearService` ispravno dohvaća `Footwear` objekt kada postoji u repository-ju.
- **Ulazne varijable**: `id = 1`
- **Očekivani rezultat**: Metoda `getById(1)` treba vratiti `Footwear` objekt s ID-jem 1 i `openness = null`.
- **Stvarni rezultat**: `Footwear` objekt nije `null`, ID je 1, `openness` je `null`, metoda `findById` repository-ja pozvana je jednom.

3.6 `testGetById_WhenFootwearDoesNotExist`

- **Opis testa**: Provjerava da li metoda `getById` u `FootwearService` ispravno rukuje slučajem kada `Footwear` objekt s danim ID-jem ne postoji.
- **Ulazne varijable**: `id = 5`

- **Očekivani rezultat**: Metoda `getById(5)` treba vratiti `null` jer `Footwear` s ID-jem 5 ne postoji.
- **Stvarni rezultat**: Metoda vraća `null`, metoda `findById` repository-ja pozvana je jednom.

3.7 `testGetAll`

- **Opis testa**: Provjerava da li metoda `getAll` u `FootwearService` ispravno vraća sve `Footwear` objekte.
- **Ulazne varijable**: Nema
- **Očekivani rezultat**: Metoda `getAll()` treba vratiti listu sa 2 `Footwear` objekta.
- **Stvarni rezultat**: Lista sa 2 `Footwear` objekta nije `null`, metoda `findAll` repository-ja pozvana je jednom.

3.8 `testDelete_WhenFootwearExists`

- **Opis testa**: Provjerava da li metoda `delete` u `FootwearService` ispravno uklanja `Footwear` objekt kada postoji u repository-ju.
- **Ulazne varijable**: `idToDelete = 1`
- **Očekivani rezultat**: Metoda `delete(1)` treba ukloniti `Footwear` objekt s ID-jem 1 i vratiti `true`.
- **Stvarni rezultat**: Metoda vraća `true`, metoda `existsById` repository-ja pozvana je jednom, metoda `articleService.remove` pozvana je jednom.

3.9 `testDelete_WhenFootwearDoesNotExist`

- **Opis testa**: Provjerava da li metoda `delete` u `FootwearService` ispravno rukuje slučajem kada `Footwear` objekt s danim ID-jem ne postoji.
- **Ulazne varijable**: `idToDelete = 6`
- **Očekivani rezultat**: Metoda `delete(6)` treba vratiti `false` jer `Footwear` s ID-jem 6 ne postoji.
- **Stvarni rezultat**: Metoda vraća `false`, metoda `existsById` repository-ja pozvana je jednom, metode `articleService.remove` nisu pozvane.

4. OrmarServiceTest

4.1 `testGet_ExistingCloset`

- **Opis testa**: Provjerava da li metoda `get` u `OrmarService` ispravno dohvaća `Closet` objekt kada postoji u repository-ju.
- **Ulazne varijable**: `closetId = 1`
- **Očekivani rezultat**: Metoda `get(1)` treba vratiti `Closet` objekt s ID-jem 1, imenom "Ormar 1" i korisničkim ID-jem 10.
- **Stvarni rezultat**: `Closet` objekt nije `null`, ID je 1, ime je "Ormar 1", korisnički ID je 10, metoda `findById` repository-ja pozvana je jednom.

4.2 `testGet_NonExistingCloset`

- **Opis testa**: Provjerava da li metoda `get` u `OrmarService` ispravno rukuje slučajem kada `Closet` objekt s danim ID-jem ne postoji.
- **Ulazne varijable**: `closetId = 999`
- **Očekivani rezultat**: Metoda `get(999)` treba vratiti `null` jer `Closet` s ID-jem 999 ne postoji.
- **Stvarni rezultat**: Metoda vraća `null`, metoda `findById` repository-ja pozvana je jednom.

4.3 `testGetByUser_WithClosets`

- **Opis testa**: Provjerava da li metoda `getByUser` u `OrmarService` ispravno dohvaća sve ormare i njihove lokacije za određenog korisnika.
- **Ulazne varijable**: `userId = 10`
- **Očekivani rezultat**: Metoda `getByUser(10)` treba vratiti mapu s imenima ormara ("Ormar 1" i "Ormar 2") mapiranim na njihove odgovarajuće liste `Location` objekata.
- **Stvarni rezultat**: Mapa nije `null`, veličina mape je 2, ključ "Ormar 1" mapira na dvije lokacije, ključ "Ormar 2" mapira na jednu lokaciju, metode `findByUserID` i `getByClosetId` pozvane su ispravno.

4.4 `testGetByUser_NoClosets`

- **Opis testa**: Provjerava da li metoda `getByUser` u `OrmarService` ispravno rukuje slučajem kada korisnik nema ormare.
- **Ulazne varijable**: `userId = 20`

- **Očekivani rezultat**: Metoda `getByUser(20)` treba vratiti praznu mapu jer korisnik nema ormare.
- **Stvarni rezultat**: Mapa nije `null`, prazna je, metode `findByUserID` pozvane su jednom, `getByClosetId` nije pozvana.

4.5 `testGetAll_WithClosets`

- **Opis testa**: Provjerava da li metoda `getAll` u `OrmarService` ispravno vraća sve `Closet` objekte.
- **Ulazne varijable**: Nema
- **Očekivani rezultat**: Metoda `getAll()` treba vratiti listu sa 3 `Closet` objekta.
- **Stvarni rezultat**: Lista sa 3 `Closet` objekta nije `null`, metoda `findAll` repository-ja pozvana je jednom.

4.6 `testGetAll_NoClosets`

- **Opis testa**: Provjerava da li metoda `getAll` u `OrmarService` ispravno rukuje slučajem kada nema ormara.
- **Ulazne varijable**: Nema
- **Očekivani rezultat**: Metoda `getAll()` treba vratiti praznu listu jer nema ormara.
- **Stvarni rezultat**: Lista nije `null`, prazna je, metoda `findAll` repository-ja pozvana je jednom.

4.7 `testGetByUserID_WithClosets`

- **Opis testa**: Provjerava da li metoda `getByUserID` u `OrmarService` ispravno dohvaća sve `Closet` objekte povezane s određenim korisnikom.
- **Ulazne varijable**: `userId = 10`
- **Očekivani rezultat**: Metoda `getByUserID(10)` treba vratiti listu sa 2 `Closet` objekta povezana s korisnikom ID 10.
- **Stvarni rezultat**: Lista sa 2 `Closet` objekta nije `null`, metode `findByUserID` repository-ja pozvana je jednom.

4.8 `testGetByUserID_NoClosets`

- **Opis testa**: Provjerava da li metoda `getByUserID` u `OrmarService` ispravno rukuje slučajem kada korisnik nema ormare.
- **Ulazne varijable**: `userId = 20`
- **Očekivani rezultat**: Metoda `getByUserID(20)` treba vratiti praznu listu jer korisnik nema ormare.

- **Stvarni rezultat**: Lista nije `null`, prazna je, metoda `findByUserID` repository-ja pozvana je jednom.

4.9 `testAdd_Closet`

- **Opis testa**: Provjerava da li metoda `add` u `OrmarService` ispravno dodaje novi `Closet` objekt.
- **Ulazne varijable**: `Closet` objekt s ID-jem 1, imenom "Ormar 1", korisničkim ID-jem 10
- **Očekivani rezultat**: Metoda `add(closet)` treba spremiti `Closet` objekt i vratiti isti objekt s ID-jem 1, imenom "Ormar 1" i korisničkim ID-jem 10.
- **Stvarni rezultat**: `Closet` objekt nije `null`, ID je 1, ime je "Ormar 1", korisnički ID je 10, metoda `save` repository-ja pozvana je jednom.

4.10 `testAdd_Closet_WithNullUserId`

- **Opis testa**: Provjerava da li metoda `add` u `OrmarService` ispravno dodaje `Closet` objekt kada korisnički ID nije postavljen.
- **Ulazne varijable**: `Closet` objekt s ID-jem 2, imenom "Ormar 2", korisnički ID je `null`
- **Očekivani rezultat**: Metoda `add(closet)` treba spremiti `Closet` objekt i vratiti isti objekt s ID-jem 2, imenom "Ormar 2" i `userid = null`.
- **Stvarni rezultat**: `Closet` objekt nije `null`, ID je 2, ime je "Ormar 2", korisnički ID je `null`, metoda `save` repository-ja pozvana je jednom.

4.11 `testAdd_WithNaziv`

- **Opis testa**: Provjerava da li metoda `add` u `OrmarService` ispravno dodaje novi `Closet` objekt koristeći samo naziv.
- **Ulazne varijable**: `naziv = "Ormar 3"`
- **Očekivani rezultat**: Metoda `add(naziv)` treba stvoriti novi `Closet` objekt s imenom "Ormar 3", spremiti ga, i vratiti spremljeni objekt s ID-jem 3 i `userid = null`.
- **Stvarni rezultat**: `Closet` objekt nije `null`, ID je 3, ime je "Ormar 3", korisnički ID je `null`, metoda `save` repository-ja pozvana je jednom s ispravno postavljenim nazivom.

4.12 `testDelete_WithLocations`

- **Opis testa**: Provjerava da li metoda `delete` u `OrmarService` ispravno uklanja `Closet` objekt i sve povezane `Location` entitete.

- **Ulazne varijable**: `closetId = 1`
- **Očekivani rezultat**: Metoda `delete(1)` treba ukloniti `Closet` objekt s ID-jem 1, sve povezane `Location` entitete (ID-ji 100 i 101), i vratiti `true`.
- **Stvarni rezultat**: Metoda vraća `true`, metode `findById`, `getByClosetId`, `delete` `Location` objekata i `delete` `Closet` objekta pozvane su ispravno.

4.13 `testDelete_NoLocations`

- **Opis testa**: Provjerava da li metoda `delete` u `OrmarService` ispravno uklanja `Closet` objekt kada nema povezanih `Location` entiteta.
- **Ulazne varijable**: `closetId = 2`
- **Očekivani rezultat**: Metoda `delete(2)` treba ukloniti `Closet` objekt s ID-jem 2 bez brisanja `Location` entiteta i vratiti `true`.
- **Stvarni rezultat**: Metoda vraća `true`, metode `findById`, `getByClosetId` i `delete` `Closet` objekta pozvane su ispravno, metode za brisanje `Location` objekata nisu pozvane.

4.14 `testDelete_NonExistingCloset`

- **Opis testa**: Provjerava da li metoda `delete` u `OrmarService` ispravno rukuje slučajem kada `Closet` objekt s danim ID-jem ne postoji.
- **Ulazne varijable**: `closetId = 999`
- **Očekivani rezultat**: Metoda `delete(999)` treba vratiti `false` jer `Closet` s ID-jem 999 ne postoji.
- **Stvarni rezultat**: Metoda vraća `false`, metode `findById`, `getByClosetId`, `delete` `Location` objekata i `delete` `Closet` objekta nisu pozvane.

4.15 `testAdd_WithExistingClosetName`

- **Opis testa**: Provjerava da li metoda `add` u `OrmarService` ispravno dodaje `Closet` objekt čak i kada naziv ormara već postoji.
- **Ulazne varijable**: `naziv = "Ormar 1"`
- **Očekivani rezultat**: Metoda `add(naziv)` treba dodati `Closet` objekt s istim nazivom, vratiti spremljeni objekt s novim ID-jem 4 i korisničkim ID-jem 30.
- **Stvarni rezultat**: `Closet` objekt nije `null`, ID je 4, ime je "Ormar 1", korisnički ID nije `null`, metoda `save` repository-ja pozvana je jednom.

4.16 `testDelete_WithNullCloset`

- **Opis testa**: Provjerava da li metoda `delete` u `OrmarService` ispravno rukuje slučajem kada je `Closet` objekt `null`.

- **Ulazne varijable**: `closetId = 3`
- **Očekivani rezultat**: Metoda `delete(3)` treba vratiti `false` jer je `Closet` objekt `null`.
- **Stvarni rezultat**: Metoda vraća `false`, metode `findById`, `getByClosetId`, `delete Location` objekata i `delete Closet` objekta nisu pozvane.

5. UserServiceTest

5.1 `testGetUser_WhenUserExists`

- **Opis testa**: Provjerava da li metoda `get` u `UserService` ispravno dohvaća `Users` objekt kada postoji u repository-ju.
- **Ulazne varijable**: `id = 1`
- **Očekivani rezultat**: Metoda `get(1)` treba vratiti `Users` objekt s ID-jem 1, korisničkim imenom "John Doe", emailom i lozinkom.
- **Stvarni rezultat**: `Users` objekt nije `null`, korisničko ime je "John Doe", metoda `findById` repository-ja pozvana je jednom.

5.2 `testGetUser_WhenUserDoesNotExist`

- **Opis testa**: Provjerava da li metoda `get` u `UserService` ispravno rukuje slučajem kada `Users` objekt s danim ID-jem ne postoji.

- **Ulazne varijable**: `id = 3`
- **Očekivani rezultat**: Metoda `get(3)` treba vratiti `null` jer `Users` s ID-jem 3 ne postoji.
- **Stvarni rezultat**: Metoda vraća `null`, metoda `findById` repository-ja pozvana je jednom.

5.3 `testGetByEmail_WhenUsersExist`

- **Opis testa**: Provjerava da li metoda `getByEmail` u `UserService` ispravno dohvaća `Users` objekte na temelju email adrese.
- **Ulazne varijable**:
  - `email = "john.doe@example.com"`
  - `email = "jane.smith@example.com"`
- **Očekivani rezultat**: Metoda `getByEmail` treba vratiti listu s odgovarajućim `Users` objektima za svaki email.
- **Stvarni rezultat**: Metoda vraća liste s pojedinačnim `Users` objektima za svaki email, metode `findByEmail` repository-ja pozvane su jednom za svaki email.

5.4 `testGetByEmail_WhenNoUsersExist`

- **Opis testa**: Provjerava da li metoda `getByEmail` u `UserService` ispravno rukuje slučajem kada nema korisnika s danom email adresom.
- **Ulazne varijable**: `email = "nonexistent@example.com"`
- **Očekivani rezultat**: Metoda `getByEmail("nonexistent@example.com")` treba vratiti praznu listu jer nema korisnika s tim emailom.
- **Stvarni rezultat**: Metoda vraća praznu listu, metoda `findByEmail` repository-ja pozvana je jednom.

5.5 `testGetAllUsers`

- **Opis testa**: Provjerava da li metoda `getAllUsers` u `UserService` ispravno vraća sve `Users` objekte.
- **Ulazne varijable**: Nema
- **Očekivani rezultat**: Metoda `getAllUsers()` treba vratiti listu sa 2 `Users` objekta.
- **Stvarni rezultat**: Lista sa 2 `Users` objekta nije `null`, metoda `findAll` repository-ja pozvana je jednom.

5.6 `testAddUser_SellerRole`

- **Opis testa**: Provjerava da li metoda `add` u `UserService` ispravno dodaje novog korisnika s ulogom "seller".
- **Ulazne varijable**:
  - `Users` objekt s korisničkim imenom "Alice Brown", emailom i lozinkom
  - `role = "seller"`
- **Očekivani rezultat**: Metoda `add(newUser, "seller")` treba spremiti `Users` objekt, dodati `Seller` entitet povezan s korisnikom ID-jem 3, i vratiti `Users` objekt s ID-jem 3.
- **Stvarni rezultat**: `Users` objekt nije `null`, ID je 3, korisničko ime je "Alice Brown", metode `save` repository-ja i `sellerService.add` pozvane su jednom, `Seller` entitet je ispravno povezan s korisnikom.

5.7 `testAddUser_RegisteredUserRole`

- **Opis testa**: Provjerava da li metoda `add` u `UserService` ispravno dodaje novog korisnika s ulogom "registereduser".
- **Ulazne varijable**:
  - `Users` objekt s korisničkim imenom "Bob Green", emailom i lozinkom
  - `role = "registereduser"`
- **Očekivani rezultat**: Metoda `add(newUser, "registereduser")` treba spremiti `Users` objekt, dodati `Registereduser` entitet s geolokacijom "Europe" povezan s korisnikom ID-jem 4, i vratiti `Users` objekt s ID-jem 4.
- **Stvarni rezultat**: `Users` objekt nije `null`, ID je 4, korisničko ime je "Bob Green", metode `save` repository-ja i `registereduserService.add` pozvane su jednom, `Registereduser` entitet je ispravno povezan s korisnikom.

5.8 `testAddRegisteredUser`

- **Opis testa**: Provjerava da li metoda `addRegistered` u `UserService` ispravno dodaje novog registrovanog korisnika s geolokacijom.
- **Ulazne varijable**:
  - `Users` objekt s korisničkim imenom "Charlie White", emailom i lozinkom
  - `geolocation = "North America"`
- **Očekivani rezultat**: Metoda `addRegistered(newUser, "North America")` treba spremiti `Users` objekt, dodati `Registereduser` entitet s geolokacijom "North America" povezan s korisnikom ID-jem 5, i vratiti `Users` objekt s ID-jem 5.
- **Stvarni rezultat**: `Users` objekt nije `null`, ID je 5, korisničko ime je "Charlie White", metode `save` repository-ja i `registereduserService.add` pozvane su

jednom, `Registereduser` entitet je ispravno postavljen s geolokacijom "North America".

5.9 `testAddSeller`

- **Opis testa**: Provjerava da li metoda `addSeller` u `UserService` ispravno dodaje novog prodavača s logom.
- **Ulazne varijable**:
  - `Users` objekt s korisničkim imenom "Diana Black", emailom i lozinkom
  - `image = "image.png"`
- **Očekivani rezultat**: Metoda `addSeller(newUser, "image.png")` treba spremiti `Users` objekt, dodati `Seller` entitet s logom "seller_logo.png" povezan s korisnikom ID-jem 6, i vratiti `Users` objekt s ID-jem 6.
- **Stvarni rezultat**: `Users` objekt nije `null`, ID je 6, korisničko ime je "Diana Black", metode `save` repository-ja i `sellerService.add` pozvane su jednom, `Seller` entitet je ispravno postavljen s logom "seller_logo.png".

5.10 `testDeleteUser_WhenUserExists`

- **Opis testa**: Provjerava da li metoda `delete` u `UserService` ispravno uklanja postojećeg korisnika i sve povezane entitete (ormari, artikle, seller, registereduser).
- **Ulazne varijable**: `userId = 1`
- **Očekivani rezultat**: Metoda `delete(userId)` treba ukloniti sve povezane ormari (ID-ji 10 i 11), artikle (ID-ji 100 i 101), `Seller` i `Registereduser` entitete povezana s korisnikom ID-jem 1, te samog korisnika, i izvršiti sve brisanja uspješno.
- **Stvarni rezultat**: Sve povezane metode za dohvaćanje i brisanje ormara, artikala, `Seller` i `Registereduser` entiteta pozvane su ispravno, `deleteById` metoda repository-ja pozvana je jednom.

5.11 `testDeleteUser_WhenUserDoesNotExist`

- **Opis testa**: Provjerava da li metoda `delete` u `UserService` ispravno rukuje slučajem kada korisnik ne postoji (bez povezanih entiteta).
- **Ulazne varijable**: `userId = 2`
- **Očekivani rezultat**: Metoda `delete(userId)` treba ukloniti korisnika s ID-jem 2 čak i ako nema povezanih ormara ili artikala, te vratiti `false` jer korisnik ne postoji.

- **Stvarni rezultat**: Metoda izvršava `deleteById` repository-ja, metode za brisanje ormara, artikala, `Seller` i `Registereduser` entiteta nisu pozvane jer korisnik ne postoji.

5.12 `testGetUserFromSession_WithValidSession`

- **Opis testa**: Provjerava da li metoda `getUserFromSession` u `UserService` ispravno dohvaća korisnički ID iz valjane sesije.
- **Ulazne varijable**: `session` objekt s atributom `sif_korisnika = 1`
- **Očekivani rezultat**: Metoda `getUserFromSession(session)` treba vratiti `Integer` vrijednost `1`.
- **Stvarni rezultat**: Metoda vraća `1`, metoda `getAttribute("sif_korisnika")` pozvana je jednom.

5.13 `testGetUserFromSession_WithInvalidSession`

- **Opis testa**: Provjerava da li metoda `getUserFromSession` u `UserService` ispravno rukuje slučajem kada sesija ne sadrži korisnički ID.
- **Ulazne varijable**: `session` objekt bez atributa `sif_korisnika`
- **Očekivani rezultat**: Metoda `getUserFromSession(session)` treba vratiti `null` jer korisnički ID nije prisutan.
- **Stvarni rezultat**: Metoda vraća `null`, metoda `getAttribute("sif_korisnika")` pozvana je jednom.

5.14 `testGetUserFromSession_WithNullSession`

- **Opis testa**: Provjerava da li metoda `getUserFromSession` u `UserService` ispravno rukuje slučajem kada je sesija `null`.
- **Ulazne varijable**: `session = null`
- **Očekivani rezultat**: Metoda `getUserFromSession(null)` treba vratiti `null` jer sesija nije dostupna.
- **Stvarni rezultat**: Metoda vraća `null`.

5.15 `testAddUser_WithUnknownRole`

- **Opis testa**: Provjerava da li metoda `add` u `UserService` ispravno dodaje korisnika kada je zadana nepoznata uloga.
- **Ulazne varijable**:
  - `Users` objekt s korisničkim imenom "Eve White", emailom i lozinkom
  - `role = "unknown_role"`

- **Očekivani rezultat**: Metoda `add(newUser, "unknown_role")` treba spremiti `Users` objekt bez dodavanja `Seller` ili `Registereduser` entiteta i vratiti `Users` objekt s ID-jem 7.
- **Stvarni rezultat**: `Users` objekt nije `null`, ID je 7, korisničko ime je "Eve White", metode `save` repository-ja pozvana je jednom, metode `sellerService.add` i `registereduserService.add` nisu pozvane.

5.16 `testAddRegisteredUser_WhenSaveThrowsException`

- **Opis testa**: Provjerava da li metoda `addRegistered` u `UserService` ispravno rukuje izuzetkom koji se baca pri spremanju `Users` objekta.
- **Ulazne varijable**:
  - `Users` objekt s korisničkim imenom "Frank Brown", emailom i lozinkom
  - `geolocation = "Asia"`
- **Očekivani rezultat**: Metoda `addRegistered(newUser, "Asia")` treba baciti `RuntimeException` s porukom "Database error" zbog greške pri spremanju.
- **Stvarni rezultat**: Metoda baca `RuntimeException` s porukom "Database error", metoda `save` repository-ja pozvana je jednom, metoda `registereduserService.add` nije pozvana.

5.17 `testDeleteUser_WithOnlySeller`

- **Opis testa**: Provjerava da li metoda `delete` u `UserService` ispravno uklanja korisnika koji ima samo `Seller` entitet, bez ormara ili artikala.
- **Ulazne varijable**: `userId = 3`
- **Očekivani rezultat**: Metoda `delete(userId)` treba ukloniti `Seller` entitet povezan s korisnikom ID-jem 3 i potom ukloniti samog korisnika, te vratiti `true`.
- **Stvarni rezultat**: Metoda izvršava `sellerService.delete(userId)` i `deleteById` repository-ja pozvane su jednom, metode za brisanje ormara i artikala nisu pozvane.

5.18 `testDeleteUser_WithOnlyRegisteredUser`

- **Opis testa**: Provjerava da li metoda `delete` u `UserService` ispravno uklanja korisnika koji ima samo `Registereduser` entitet, bez ormara ili artikala.
- **Ulazne varijable**: `userId = 4`
- **Očekivani rezultat**: Metoda `delete(userId)` treba ukloniti `Registereduser` entitet povezan s korisnikom ID-jem 4 i potom ukloniti samog korisnika, te vratiti `true`.

- **Stvarni rezultat**: Metoda izvršava `registereduserService.delete(userId)` i `deleteById` repository-ja pozvane su jednom, metode za brisanje ormara i artikala nisu pozvane.