NoSQL

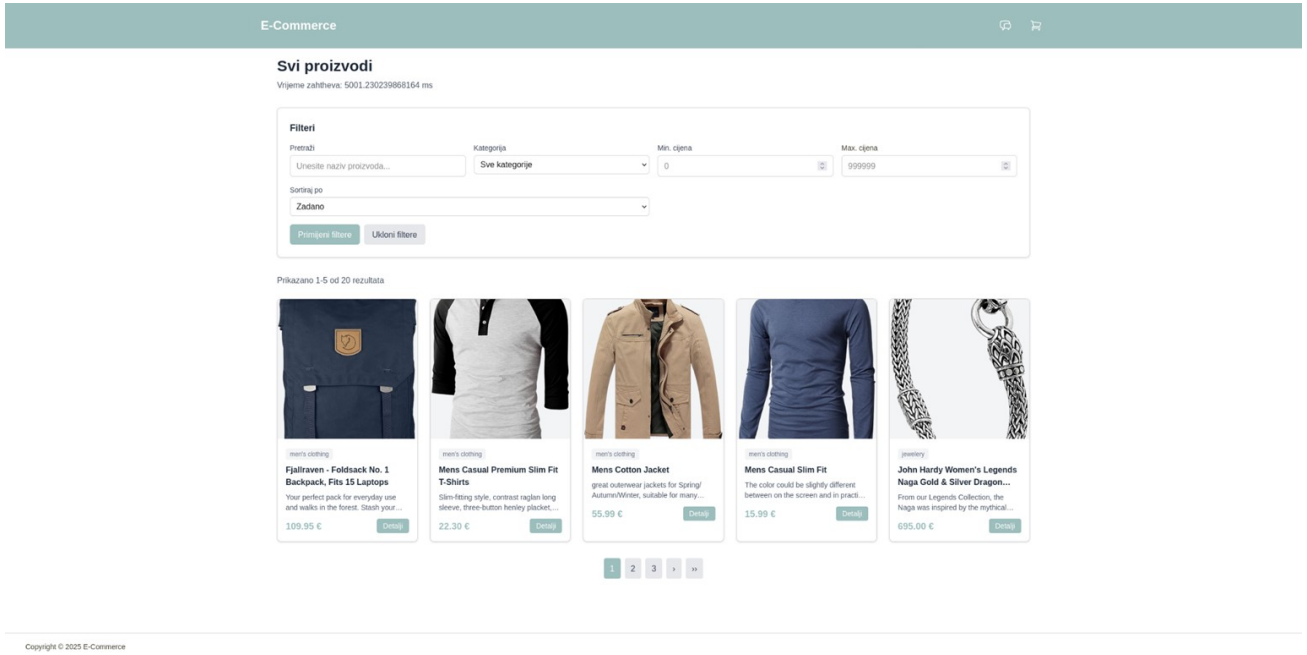
FERIT

Primjena Redisa

Laboratorijska vježba 2

Opis vježbe

- Primjena Redis alata u primjeru web shopa
- Cachiranje stranica
- Upravljanje sesijama
- Upravljanje košaricom
- Chat sustav za korisničku podršku



Postavljanje okruženja

1. Kreirajte direktorij na vašem računalu za ovu vježbu
2. Unutar tog direktorija, klonirajte repozitorij https://gitlab.com/jurajperic/lv_2_e_commerce
3. Otvorite klonirani projekt u Visual Studio Code okruženju
4. Otvorite terminal i pozicionirajte se u direktorij projekta
5. Pokrenite Django server, Redis server i Redis Insight naredbom: `docker-compose up -d`
6. Provjerite stranicu u pregledniku na localhost:8000
7. Spojite bazu u Redis Insight u novom tabu

Zadatak 1: Cacheiranje stranice

- **Problem:**
  - Webshop s velikim brojem proizvoda
  - Pretrage s filterima mogu dugo trajati
  - Korisnici često koriste slične filtere
  - Nepotrebno opterećenje sustava istim zahtjevima

- **Rješenje**:
  - Cacheiranje rezultata pretrage na određeni period
  - Smanjenje broja zahtjeva prema bazi podataka
  - Poboljšano korisničko iskustvo

Zadatak 1: Cacheiranje stranice

- **Postaviti cache u postavkama Django projekta**

settings.py

```
CACHES = {
    "default": {
        "BACKEND": "django.core.cache.backends.redis.RedisCache",
        "LOCATION": "redis://default:studentlab@redis:6379",
        "KEY_PREFIX": "e_commerce",

    }
}
```

https://docs.djangoproject.com/en/5.2/topics/cache/

Zadatak 1: Cacheiranje stranice

- **Cacheiranje korištenjem Django dekoratora**
  - Django ugrađeni dekorator za automatsko cachiranje cijele stranice
  - Dekorator hvata cijeli HTTP response prije nego se pošalje korisniku
  - Kreira cache ključ baziran na URL-u zahtjeva (uključujući query parametre)

- **Zadatak:**
  - Provjerite vremensku razliku u dohvaćanju stranice sa i bez cacheiranja
  - Provjerite dohvaćanje za različite kombinacije filtera
  - Provjerite podatke u Redis Insight-u

https://docs.djangoproject.com/en/5.2/topics/cache/

Zadatak 1: Cacheiranje stranice

- **Cacheiranje korištenjem low-level cache API-ja**
  - Omogućava cachiranje samo određenih dijelova podataka, ne cijele stranice
  - Granularna kontrola - možete cachirati samo skupe upite ili dijelove podataka
  - Ručno upravljanje cache ključevima i timeout vrijednostima

- **Zadatak:**
  - Cacheirati kategorije proizvoda
  - Cacheirati samo vremenski zahtjevne dijelove koda
  - Usporediti vremensku razliku sa i bez cacheiranja
  - Provjerite podatke u Redis Insight-u

https://docs.djangoproject.com/en/5.2/topics/cache/

Zadatak 2: Implementacija košarice

- **Problem:**
  - Potrebno je čuvati stanje košarice za svakog korisnika
  - Brz pristup i ažuriranje proizvoda u košarici
  - Privremeno spremanje podataka

- **Rješenje**:
  - Korištenje Django session mehanizma s Redis backendom
  - Brzo čitanje i pisanje podataka sesije
  - Automatsko čišćenje isteklih sesija

Zadatak 2: Implementacija košarice

- **Postaviti Django da koristi Redis za spremanje podataka sesije**

settings.py

```
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
```

https://docs.djangoproject.com/en/5.2/topics/http/sessions/

Zadatak 2: Implementacija košarice

- **Zadatak:**
  - Implementirati logiku za upravljanje košaricom - implementirati funkcije:
    - **cart_add**
    - **cart_update**
    - **cart_delete**
    - **cart_clear**
  - Provjeriti dostupnost proizvoda prije dodavanja/ažuriranja
  - Testirati funkcionalnost i provjeriti podatke u Redis Insight-u

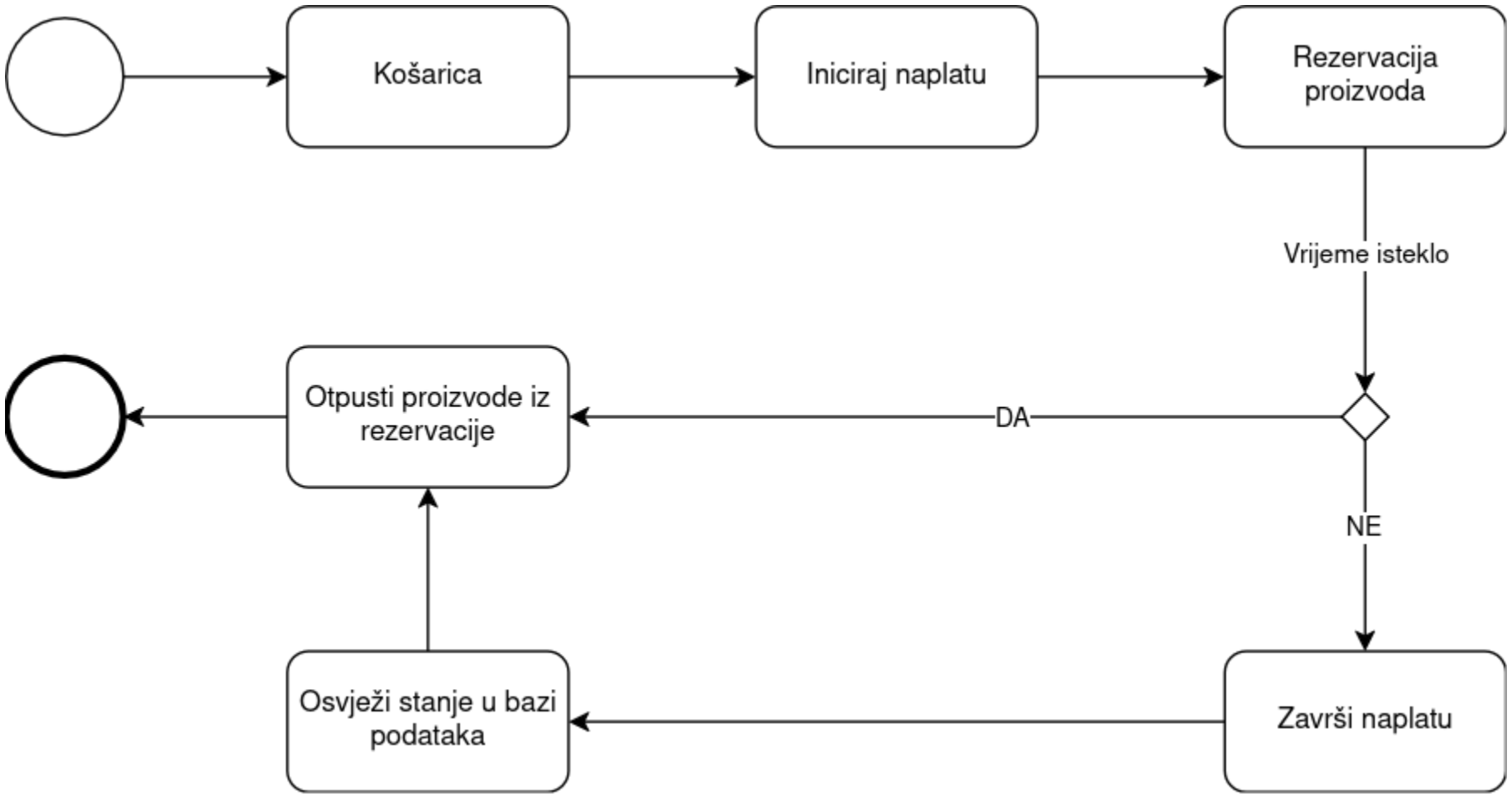
https://docs.djangoproject.com/en/5.2/topics/http/sessions/

Zadatak 3: Rezervacija proizvoda

- **Problem:**
  - Dva kupca istovremeno žele kupiti zadnji dostupan proizvod
  - Rizik od prekomjerne prodaje
  - Rizik od "zarobljavanja" proizvoda u napuštenim košaricama

- **Rješenje**:
  - Privremena rezervacija proizvoda na određeni vremenski period
  - Kupac ima dovoljno vremena za naplatu
  - U slučaju isteka rezervacija automatski ističe i proizvod se oslobađa za druge

Zadatak 3: Rezervacija proizvoda



Zadatak 3: Rezervacija proizvoda

- **RedisReservationService**
  - Servis za proces rezervacije proizvoda
  - Omogućava izravnu vezu s Redisom (koristi Redis-py biblioteku)
  - Omogućava korištenje naprednih Redis funkcionalnosti koje nisu dostupne kroz Django cache apstrakciju (npr. **pipeline**)

- **Zadatak:**
  - U **reservation_service.py** dovršiti implementacije svih funkcija u klasi **RedisReservationService**
  - Za višestruke operacije nad Redis klijentom koristiti **pipeline**
  - Koristiti funkcije **hset**, **hexpire**, **hgetall**, **hdel**, **expire**, **keys…**

Zadatak 3: Rezervacija proizvoda

- **Zadatak:**
  - U **views.py** kreirati Redisov Connection Pool koji će služiti za čuvanje veza na bazu
  - Instancirati **RedisReservationService** i predati mu jednu vezu Connection Poola
  - Dovršiti funkcije **checkout** i **complete_checkout** kako je prikazano na dijagramu
  - U funkcije **item_detail** i **cart_add** postaviti vrijednost količine proizvoda uzevši u obzir i rezervirane proizvode (**get_active_reservations_count**)
  - Testirati funkcionalnost i provjeriti podatke u Redis Insight-u

Zadatak 4: Chat

- **Problem:**
  - Potreban je sustav za razmjenu poruka u stvarnom vremenu između korisnika i podrške
  - Klasične baze podataka nisu optimizirane za brzu razmjenu poruka i visoku konkurentnost
  - Potrebno je osigurati brzu isporuku poruka bez nepotrebnog opterećenja sustava

- **Rješenje**:
  - Korištenje Redis Publish/Subscribe modela za slanje poruka u stvarnom vremenu
  - Korištenje Redis Streamova (XADD) za pohranu povijesti razgovora
  - Omogućava brzu i skalabilnu arhitekturu za chat sustav

Zadatak 4: Chat

- **Zadatak:**
  - Dodati novi ConnectionPool i Redis client u **consumers.py**
  - Nadopuniti metode u **consumers.py** pod **UserConsumer**
  - Nadopuniti metode u **consumers.py** pod **AgentConsumer**
  - U **views.py** implementirati funkciju **get_chat_streams_typed** koja dohvaća zadnjih 20 agentovih razgovora
  - U **views.py** implementirati funkciju **get_chat_stream** koja dohvaća zadnjih 20 korisnikovih poruka s agentom
  - Istestirati implementaciju
    - Otvoriti chat
    - Otvoriti novi privatni prozor u pretraživaču i upisati **localhost:8000/assistant/customer-support/** i prijaviti se
    - Istestirati razgovor između korisnika i agenta u stvarnom vremenu
    - Pogledati kako izgledaju podaci u Redis Insight-u

Kraj vježbe:

- Da biste zaustavili i obrisali Redis container, u terminalu (izvan redis-cli) upišite: **docker-compose down -v**