УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

CLOUD COMPUTING U ELEKTROENERGETSKIM SISTEMIMA

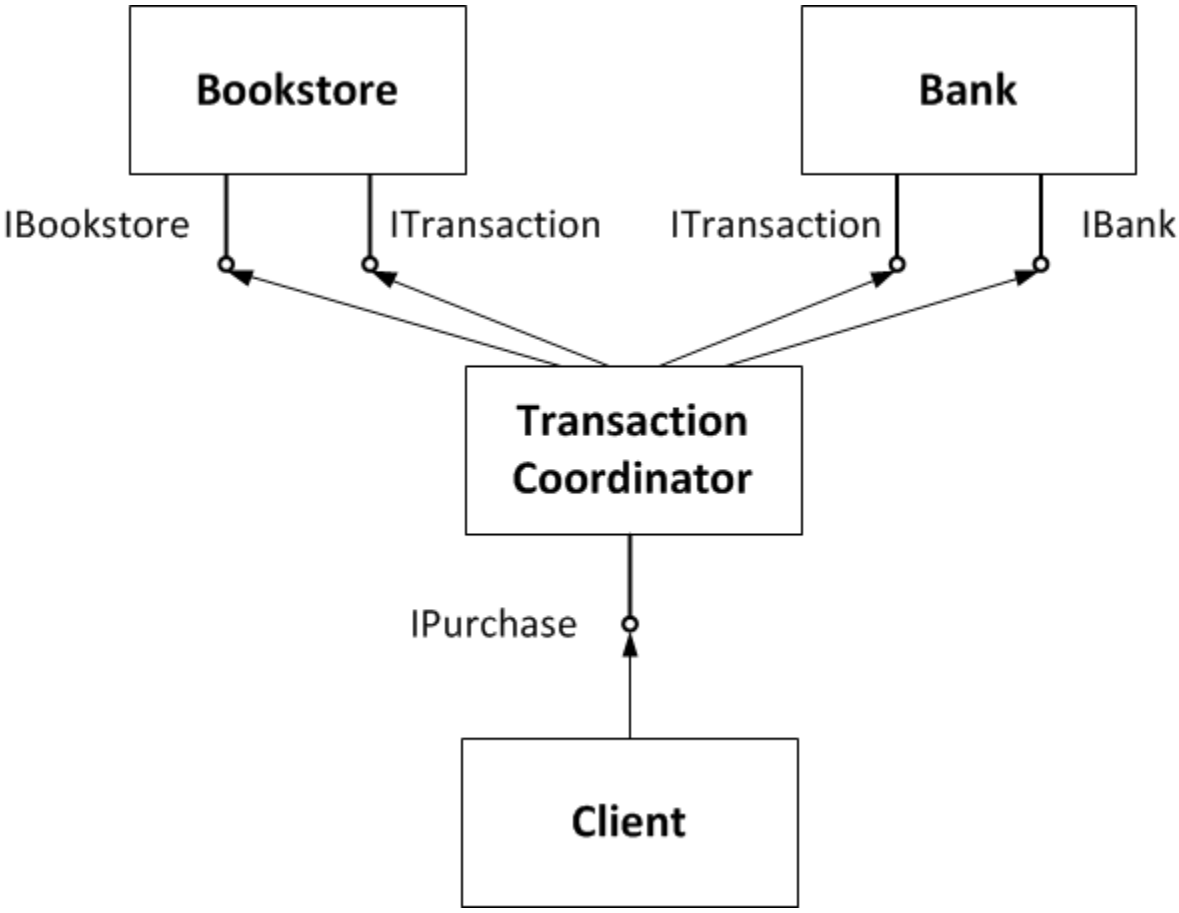
Osnove *Microsoft WindowsAzure* servisa

-SKRIPTA-

Novi Sad, 2018

Vežba 5 – Distribuirana transakcija u cloud-u

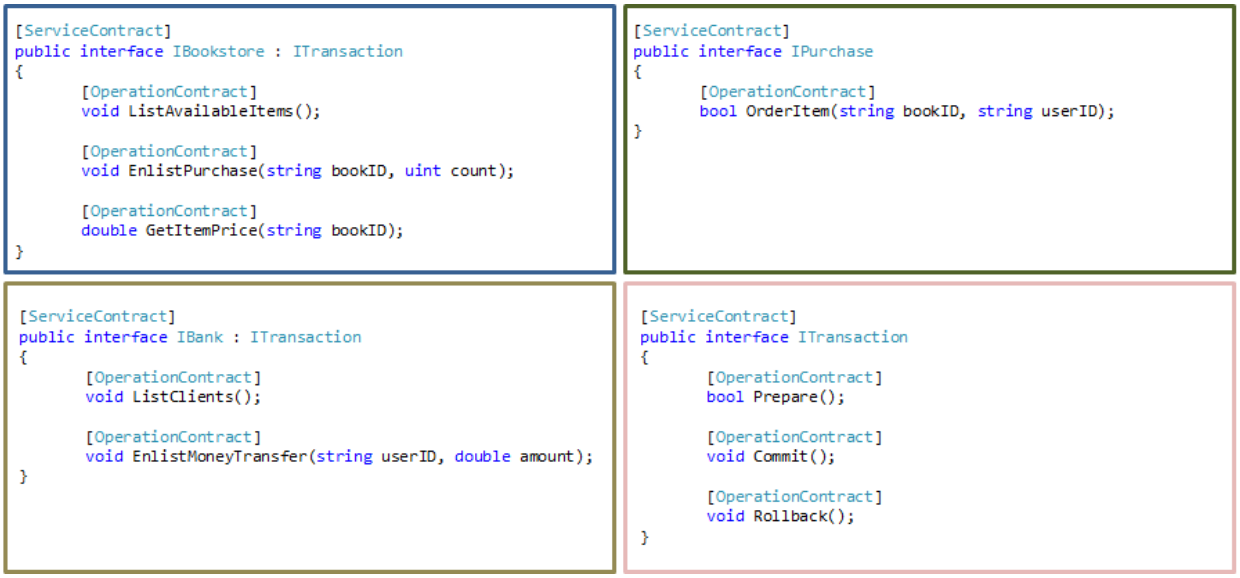
Zadatak je implementirati distribuiranu transakciju prikazanu na slici 1.



Slika 1 – šematski prikaz zadatka

Preporuka je da se zadatak sastoji iz sledećih projekata:

- DistributedTransaction (*azure cloud project*)
  - Bookstore (*worker role*)
  - Bank (*worker role*)
  - TransactionCoordinator (*worker role*)
- Client (*console application*)
- Common (*class library*) - sadrži sve interfejse prikazane na slici 2



Slika 2 – interfejsi

Bookstore je *worker role* tip procesa u *cloud* projektu koji ima implementiran *IBookstore* interfejs i u njemu će se naći implementirane sve metode iz *IBookstrore* i *ITransaction* interfejsa. Potrebno je podići WCFServis čiji je contract *IBookstore*.

- ListAvailableItems() – metoda koja ispisuje u emulator sve knjige
- EnlistPurchase(string bookID, uint count) – priprema za početak 2PC protokola, potrebno je lokalno sačuvati parameter *bookID* i *count*
- GetItemPrice(string bookID) – vraća cenu knjige čiji je ID prosleđeni *bookID*
- Prepare() – pronalazi u tabeli knjigu koja ima *bookID* koji je prosleđen *Enlist* metodom. Ukoliko je moguće smanjiti stanje knjige za vrednost *count* u tabelu se upisuje novi entitet čiji će *RowKey* biti *bookID* + "prep", taj novi entitet će imati umanjeno stanje za *count* i metoda vraća *true*. U suprotnom, metoda vraća *false*
- Commit() – Pronađe se entitet *bookID* + "prep" i njegovo stanje se prepiše u entitet *bookID*, a on se briše.
- Rollback() – Ukoliko u tabeli postoji entitet *bookID* + "prep" obriše se

Ista logika se primenjuje na *Bank* servis, samo što se implementiraju metode iz *IBank* i *ITransaction* interfejsa. Potrebno je podići WCFServis čiji je contract *IBank.*

- ListClients() – metoda koja ispisuje u emulator sve klijente
- EnlistMoneyTransfer(string userID, double amount) - priprema za početak 2PC protokola, potrebno je lokalno sačuvati parameter *userID* i *amount*

- Prepare() – pronalazi u tabeli korisnika koji ima *userID* koji je prosleđen *Enlist* metodom. Ukoliko je moguće smanjiti stanje na računu za vrednost *amount* u tabelu se upisuje novi entitet čiji će *RowKey* biti *userID* + "prep", taj novi entitet će imati umanjeno stanje računa za *amount* i metoda vraća *true*. U suprotnom, metoda vraća *false*
- Commit() – Pronađe se entitet *userID* + "prep" i njegovo stanje računa se prepiše u entitet *userID*, a on se briše.
- Rollback() – Ukoliko u tabeli postoji entitet *userID* + "prep" obriše se

Prilikom pokretanja *Bank* i *Bookstore worker rola* treba da se čuvaju sledeći podaci u Azure tabeli (može se koristiti ista tabela za oba tipa entiteta):

Bookstore:

- BookID
- Stanje knjiga na lageru
- Cena knjige

Bank servis:

- UserID
- Stanje na računu

TransactionCoordinator implementira metodu iz *IPurchase* interfejsa i uspostavlja konekciju sa *Bank* i *Bookstore* servisima. Koordinator treba da implementira two-phase commit protokol u implementaciji metode *OrderItem*. Ideja koordinatora je da započne transakciju (Enlist), proveri da li oba servisa mogu da pređu iz jednog konzistentnog stanja u drugo (Prepare), i da na osnovu toga završi transakciju (Commit ili Rollback). Prvo se pozivaju Enlist i Prepare metode i jednog i drugog servisa. Ukoliko su uslovi za prelazak u naredno stanje ispunjeni kod oba servisa, koordinator poziva Commit, a u suprotnom Rollback.

Client inicira komunikaciju sa *TransactionCoordinator* servisom putem *IPurchase* interfejsa.