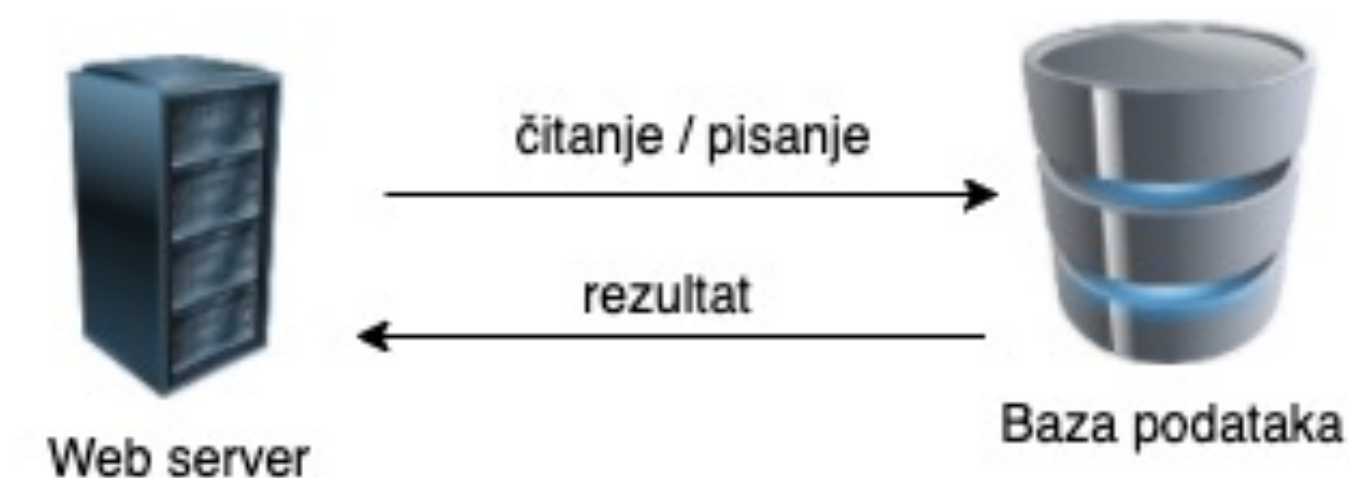
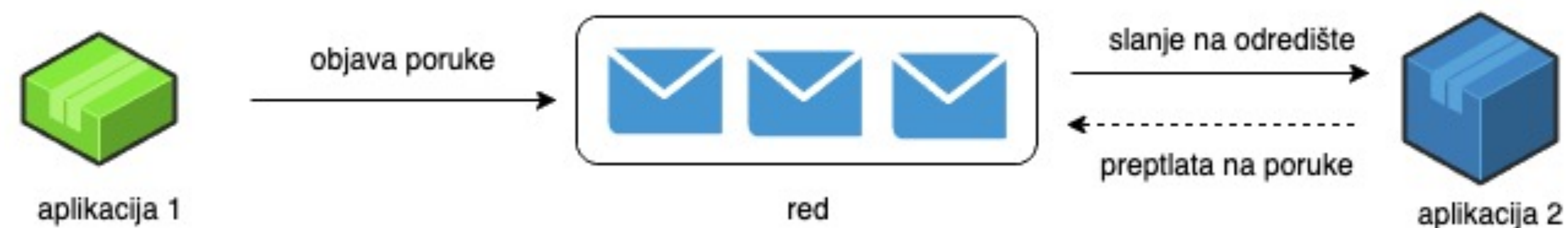
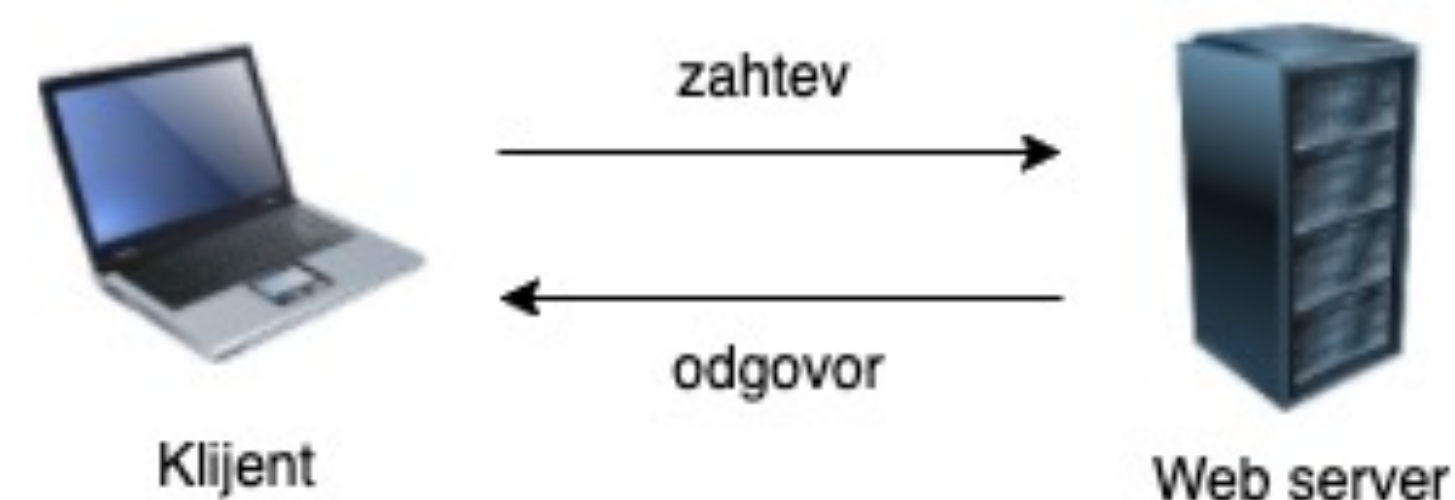


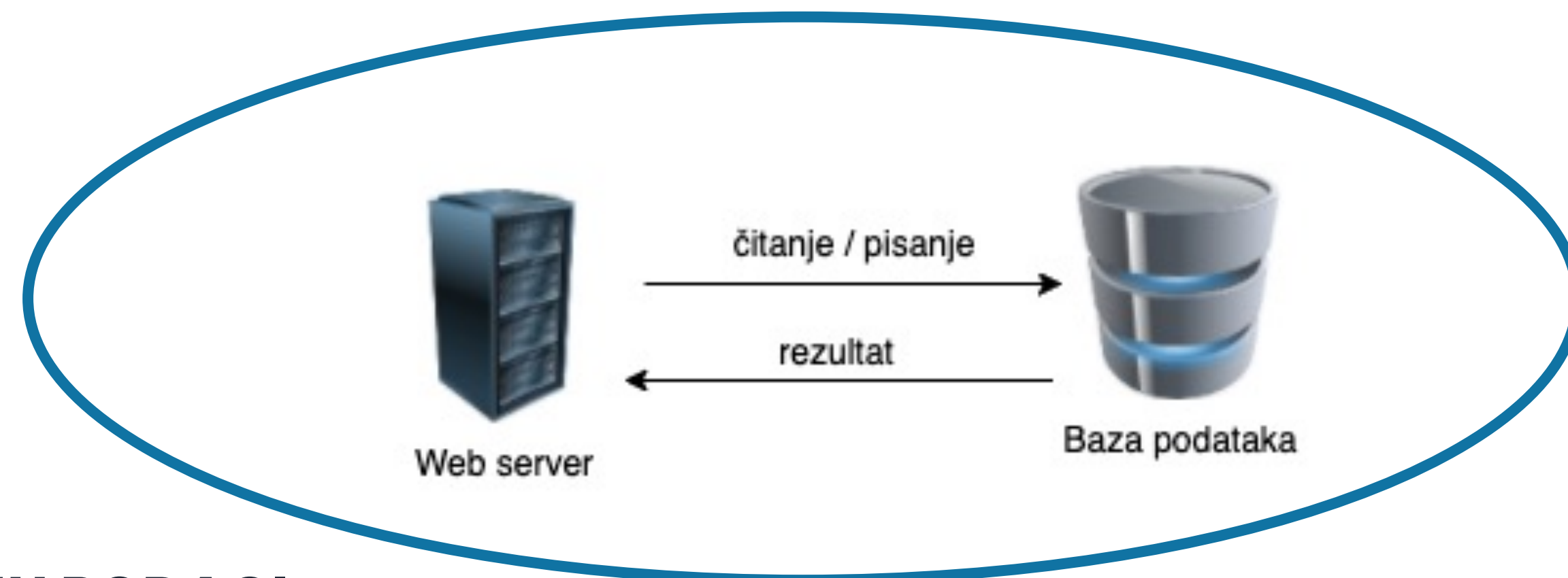
KOMUNIKACIJA SA BAZOM PODATAKA



TRI NAJČEŠĆA SCENARIJA U KOJIMA SE RAZMENJUJU PODACI

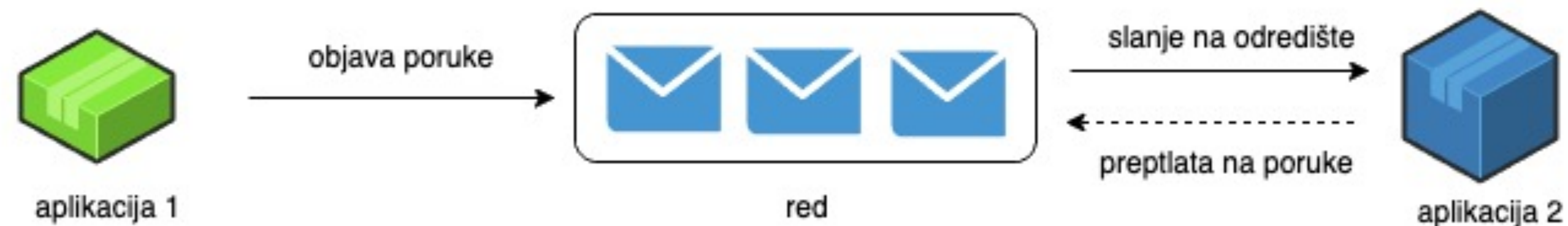
- Serverska aplikacija <-> Baza podataka
- Direktna komunikacija klijent <-> server kroz poziv servisa
- Asinhrona komunikacija razmenom poruka preko reda poruka (message queue)





TRI NAJČEŠĆA SCENARIJA U KOJIMA SE RAZMENJUJU PODACI

- Serverska aplikacija <-> Baza podataka
- Direktna komunikacija klijent <-> server kroz poziv servisa
- Asinhrona komunikacija razmenom poruka preko reda poruka (message queue)

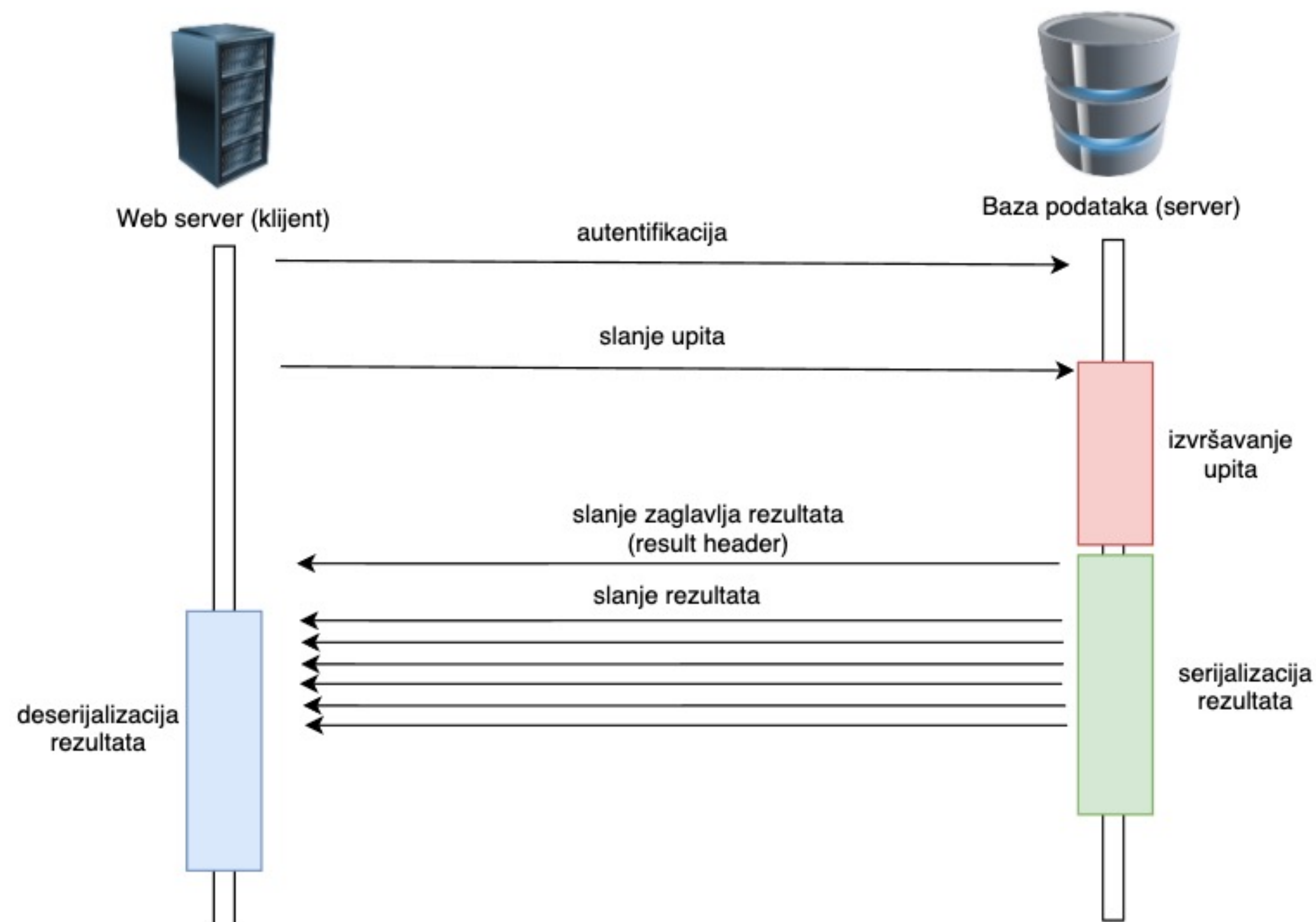




KOMUNIKACIJA SA BAZOM PODATAKA

4

◆ APLIKACIJE PRISTUPAJU BAZI PODATAKA (SUBP, DBMS) KROZ API





◆ APLIKACIJE PRISTUPAJU BAZI PODATAKA (SUBP, DBMS) KROZ API

- Direktan pristup (specifičan za konkretan DBMS)
- Open Database Connectivity (ODBC)
- Java Database Connectivity (JDBC)



OPEN DATABASE CONNECTIVITY

6

◆ ŠTA JE ODBC?

- Predstavlja standardni API za pristup DBMS
- Dizajniran da bude nezavisan od DBMS i OS
- Microsoft i Simba Technologies su ga razvili ranih 1990ih
- Svaki ozbiljniji DBMS ima ODBC implementaciju

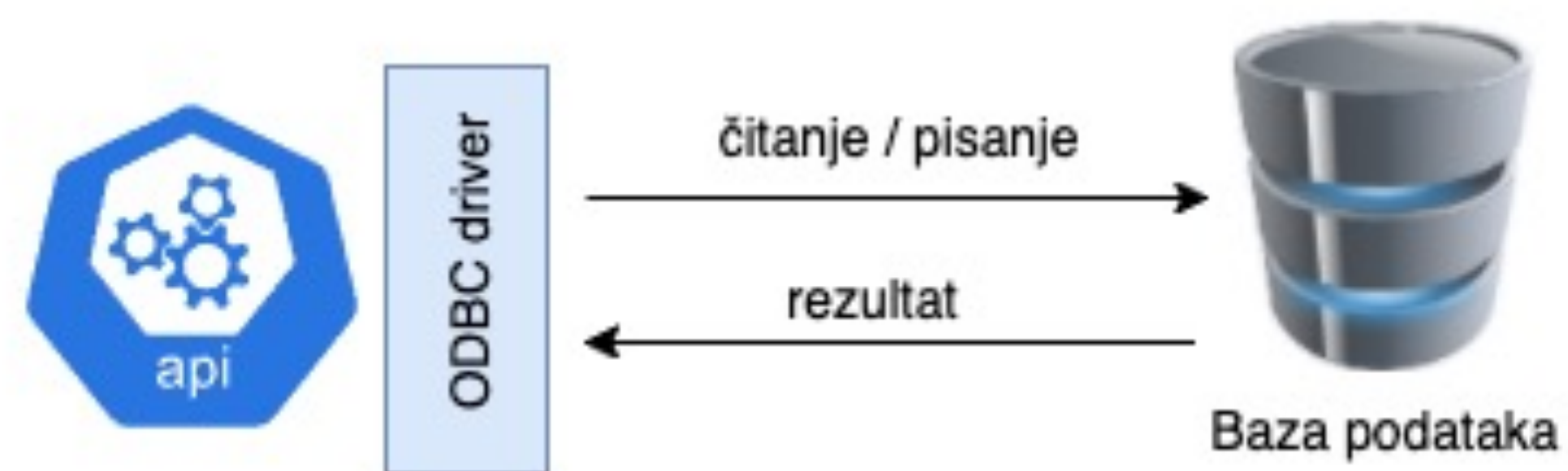


OPEN DATABASE CONNECTIVITY

7

◆ ŠTA JE ODBC?

- ODBC se bazira na „device driver“¹ modelu
- Driver enkapsulira logiku koja je potrebna za konverziju standardnog skupa komandi u DBMS specifične pozive



¹ Driver model <https://www.kernel.org/doc/html/latest/driver-api/driver-model/index.html>

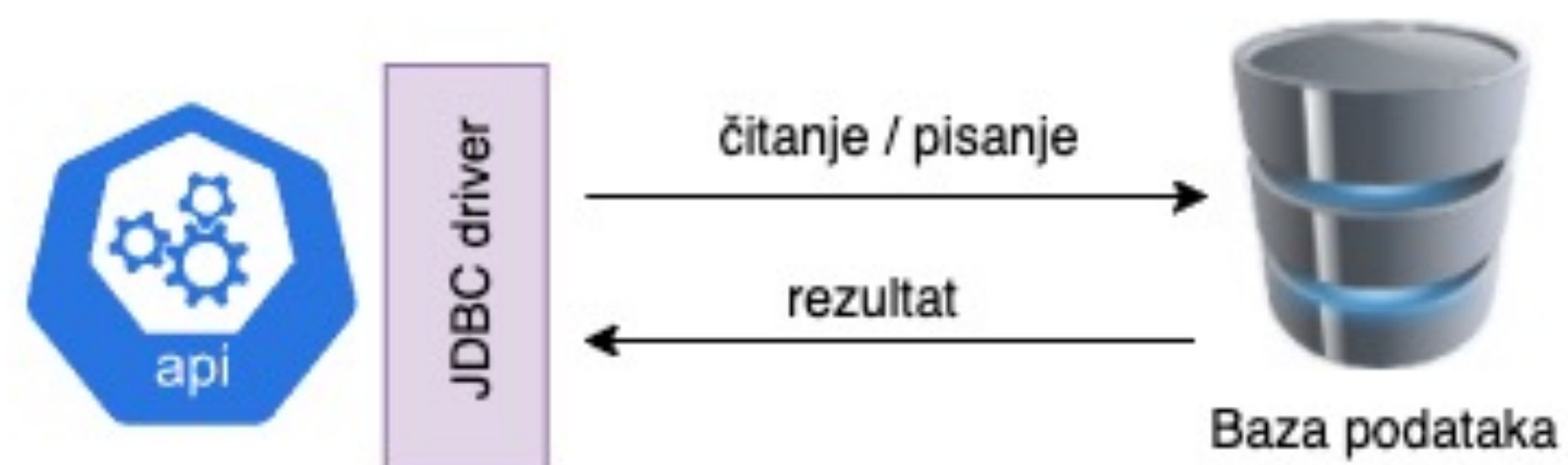


JAVA DATABASE CONNECTIVITY

8

◆ ŠTA JE JDBC?

- Sun Microsystems ga je razvio 1997.
- Pruža standardni API za konekciju Java programa sa DBMS
- Možemo ga posmatrati kao verziju ODBC pisanu u Javi umesto u C programskom jeziku





◆ PRISTUPI ZA IMPLEMENTACIJU

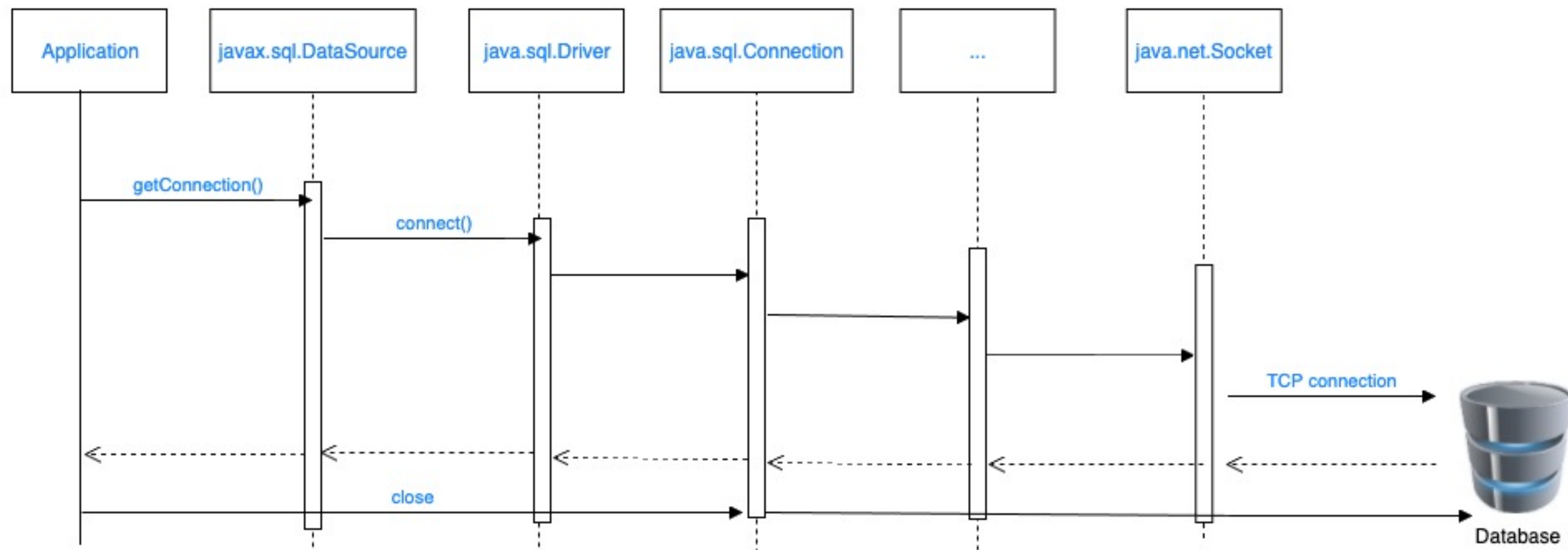
- JDBC-ODBC Bridge
 - Konvertuje JDBC poziv metode u ODBC poziv funkcije (od JDK 1.8 nije podržan pristup)
- Native-API Driver
 - Konvertuje JDBC poziv metode u nativni poziv DBMS API
- Network-Protocol Driver
 - Driver se konektuje na middleware koji konvertuje JDBC poziv u DBMS specifični poziv
- Database-Protocol Driver/Thin Driver
 - Najbrži pristup jer je čista Java implementacija koja konvertuje JDBC poziv u DBMS specifični



◆ PRISTUPI ZA IMPLEMENTACIJU

- Svi veći proizvođači DBMS implementiraju svoje mrežne protokole preko TCP/IP
- Noviji DBMS implementiraju neke od open-source DBMS mrežnih protokola
- To im omogućava da koriste postojeće drajvere bez potrebe da razvijaju i održavaju svoje

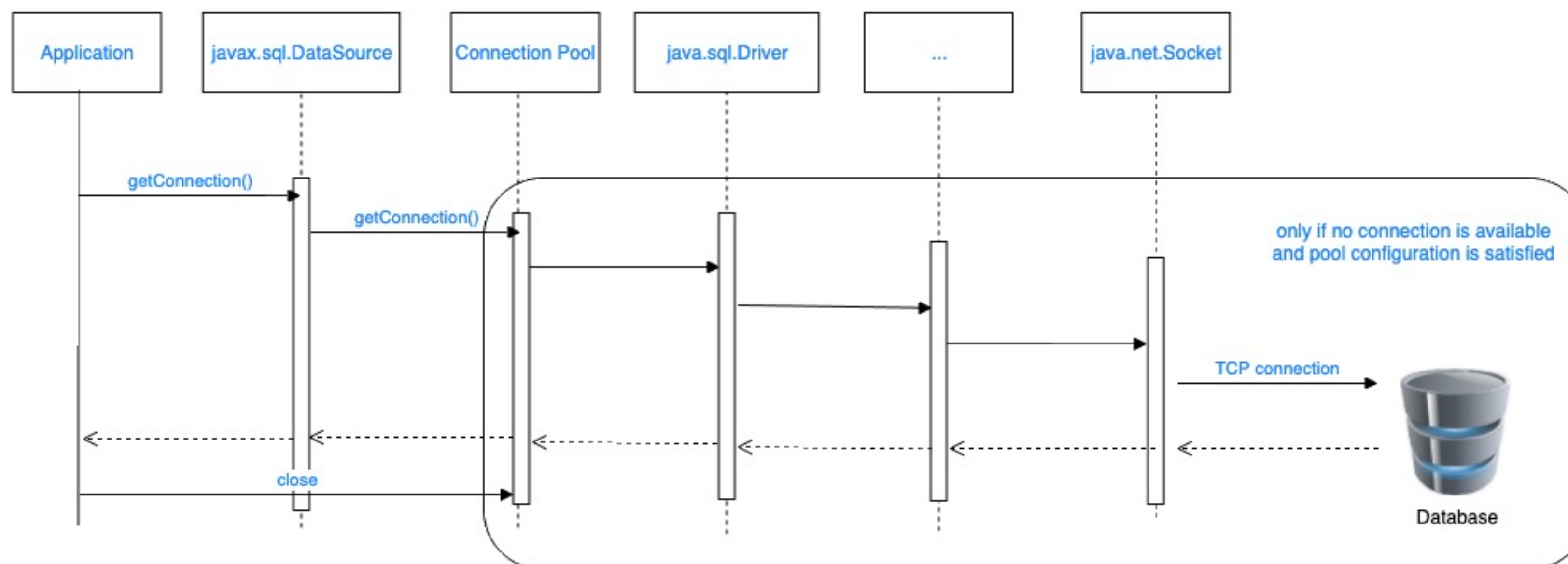
◆ KAKO SE OSTVARUJE KONEKCIJA SA BAZOM?



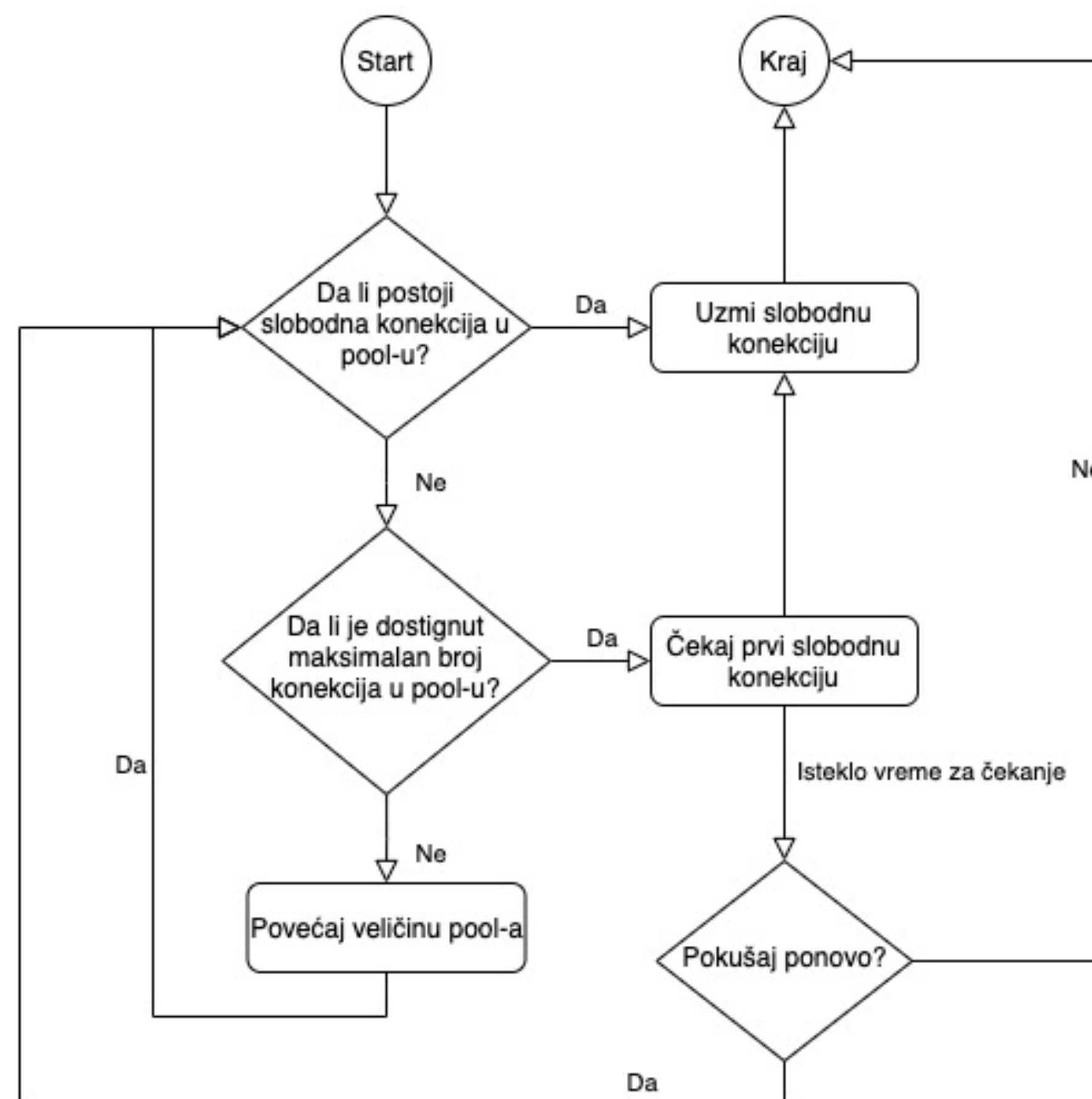
CONNECTION POOL

◆ KAKO DA SMANJIMO I/O OVERHEAD IZMEĐU APLIKACIJE I DBMS?

- Korišćenjem connection pool-a koji predstavlja keš za konekcije koje se mogu iznova koristiti bez potrebe kreiranja novih



◆ KAKO IZGLEDA LOGIKA PRIBAVLJANJA KONEKCIJE?





CONNECTION POOL

14

◆ POPULARNE IMPLEMENTACIJE

- Hikari-CP¹
- C3P0²
- Tomcat-JDBC³
- Apache common DBCP⁴

¹ <https://github.com/brettwooldridge/HikariCP>

² <https://www.mchange.com/projects/c3p0/>

³ <http://tomcat.apache.org/tomcat-7.0-doc/jdbc-pool.html>

⁴ <http://commons.apache.org/proper/commons-dbcp/>



REFERENCE

15

- ◆ **PRIMERI PO UZORU NA** <https://github.com/mbranko/isa19/tree/master/06-pooling>
- ◆ **PAVLO A., CMU. ADVANCED DATABASE SYSTEMS – NETWORKING**
<https://15721.courses.cs.cmu.edu/spring2020/slides/11-networking.pdf>
- ◆ **RAASVELDT M., MUHLEISEN H., DON'T HOLD MY DATA HOSTAGE – A CASE FOR CLIENT PROTOCOL REDESIGN**
<https://15721.courses.cs.cmu.edu/spring2020/papers/11-networking/p1022-muehleisen.pdf>
- ◆ **MIHALCHEA V. THE ANATOMY OF CONNECTION POOLING**
<https://vladmihalcea.com/the-anatomy-of-connection-pooling/>
- ◆ **HIKARI-CP. ABOUT POOL SIZING** <https://github.com/brettwooldridge/HikariCP/wiki/About-Pool-Sizing>

**KOJA SU VAŠA
PITANJA?**