Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Programska potpora komunikacijskim sustavima – završni ispit

Ime i prezime:					
JMBAG:		Akad. godina:	2022./2023.	Datum:	28. 6. 2023.

Izjavljujem da tijekom izrade ove zadaće neću od drugoga primiti niti drugome pružiti pomoć, te da se neću koristiti nedopuštenim sredstvima. Ove su radnje teška povreda Kodeksa ponašanja te mogu uzrokovati i trajno isključenje s Fakulteta. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće.

Vlastoručni potpis: _____

Napomena: U zagradama ( ) su bodovi koji se mogu ostvariti točnim odgovorom na pitanje. Nema negativnih bodova. Trajanje: 90 minuta.

PITANJA

1. (3 boda) Trebate napraviti pretragu na web stranici https://www.fer.unizg.hr/. Kako bi mogao izgledati URL za, primjerice, upit "*ppks*"? Koju metodu protokola HTTP biste koristili?

ODGOVOR: http://www.fer.unizg.hr/search?q=ppks A METODA BI BILA GET

2. (4 boda) Nadopunite klasu PersonController tako da na mjestu označenom sa znakom @ kojim počinju anotacije nadopišete odgovarajuću iz sljedeće liste mogućih anotacija: `@Autowired`, `@DeleteMapping`, `@GetMapping`, `@PostMapping`, `@PathVariable`, `@PutMapping`, `@RequestBody`, `@RestController`. Napomena: potrebno je iskoristiti sve anotacije, a neke od anotacija se mogu pojaviti više puta.

ODGOVORI U KODU

```
@ RestController
public class PersonController {

    @ AutoWired
    private PersonService personService;

    @ GetMapping ("/persons")
    public List<Person> get() {
        return personService.get();
    }

    @ GetMapping ("/persons/{id}")
    public Person get(@ PathVariable ("id") long id) {
        return personService.get(id);
    }

    @ PostMapping ("/persons")
    public void add(@ RequestBody Person person) {
        personService.add(person);
    }

    @ DeleteMapping ("/persons/{id}")
    public void delete(@ PathVariable ("id") long id) {
        personService.delete(id);
    }

    @ PutMapping ("/persons/{id}")
    public void update(@ RequestBody Person person, @ PathVariable ("id") long id) {
        personService.update(person, id);
    }

}
```

3. (1 bod) Koju naredbu bi trebali koristiti kako bi ispisivali log izvođenja Javascripta u konzolu?

ODGOVOR: console.log("Poruka");

4. (3 boda) Navedite 3 (od ukupno 5) ograničenja arhitekturnog stila REST i svaki objasnite jednom rečenicom.

ODGOVOR: model klijent server - jasna podjela odgovornosti gdje ne rade svi sve a odgovornost korisnickog sucelja i pohrane podataka su odvojene.
nema stanja - posluzitelj ne pamti komunikacije s klijentom pa je svaki zahtjev neovisan
slojevitost posluzitelja - izmedu posluzitelja i klijenta se moraju moci nalaziti posluziteljske komponente kao sto su zastupnik ili uravnotezivac opterecenja

5. (2 boda) Objasnite što radi sljedeća metoda.

```
@Override
public Person get(long id) {
    try {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(schemeHostPort + "/persons/" + id))
                .GET()
                .build();
        HttpResponse response = httpClient.send(request, BodyHandlers.ofString());
        Person person = (Person) Person.fromJson(response.body().toString());
        return (response.statusCode() == 200) ? person : null;
    } catch (IOException | InterruptedException ex) {
        return null;
    }
}
```

ODGOVOR: Ova metoda šalje HTTP GET zahtjev na određeni URL kako bi dohvatila podatke osobe s određenim ID-om. Ako odgovor ima status kod 200, tumači odgovor kao JSON i pretvara ga u objekt klase Person te ga vraća. U suprotnom, vraća null. U slučaju bilo kakve greške tijekom izvođenja zahtjeva ili parsiranja odgovora, također vraća null.

6. *(4 boda) Ispravnim redoslijedom poredajte metode na strani TCP poslužitelja i klijenta tako što ćete pored svake metode staviti njen redni broj. Napomena: napravite to odvojeno za klijenta i poslužitelja tj. neka redni brojevi od 1. do 8. budu za poslužitelja, a redni brojevi od 1. do 5. za klijenta.*

Poslužitelj	Klijent
accept()	close()
bind()	connect()
close()	read()
listen()	socket()
read()	write()
socket()	
socket()	
write()	

ODGOVOR: Poslužitelj: socket() bind() listen() accept() read() write() close()
Klijent: socket() connect() read() write() close()

7. (4 boda) Ukratko objasnite najvažnije razlike između protokola UDP i SCTP.

ODGOVOR: UDP je nepouzdan protokol koji ne garantira isporuku paketa ili redoslijed, nema ugrađenu kontrolu toka ni mehanizme za ispravljanje grešaka, te ne podržava višestruke tokove podataka unutar jedne sesije. SCTP je pouzdan protokol koji osigurava isporuku paketa u ispravnom redoslijedu i bez duplikata, ima ugrađene mehanizme za kontrolu toka i ispravljanje grešaka, te podržava višestruke tokove podataka unutar jedne sesije.

8. (3 boda) Što znači da kod protokola SCTP svaka asocijacija podržava više logičkih tokova poruka? Označite mjesto u sljedećem isječku programskog koda na kojem se definira logički tok kojim se šalju podaci.

```
20, "C");     Reading reading = new Reading("id_4", "temperature", (Math.random() * 60) -

              byte[] message = reading.toJson().getBytes("UTF-8");
              ByteBuffer byteBuffer = ByteBuffer.wrap(message);
              System.out.println("Sent: " + reading);
              MessageInfo messageInfo = MessageInfo.createOutgoing(null, 0);
              clientSctpChannel.send(byteBuffer, messageInfo); //SEND
```

ODGOVOR: Kod koji se odnosi na slanje poruke ne sadrži eksplicitnu definiciju logičkog toka. Međutim, kod protokola SCTP, logički tok obično nije izričito definiran na ovom nivou, već se obično upravlja višim nivoima u aplikaciji. Kod koji je dat koristi clientSctpChannel za slanje podataka, ali ne specificira logički tok. U protokolu SCTP, logički tokovi su obično vezani uz veze (association) i nisu eksplicitno definirani na ovom nivou koda. Dakle, definicija logičkog toka obično bi se nalazila na višem nivou, prilikom uspostavljanja veze ili nekog drugog upravljanja tokovima poruka.

9. (2 boda) Kreirate bazu podataka s podacima korisnika. Koji tip podataka biste koristili za ime korisnika?

ODGOVOR: Za polje koje će sadržavati ime korisnika u bazi podataka, najčešće se koristi tip podataka VARCHAR. Ovaj tip podataka omogućuje pohranu varijabilne duljine niza znakova, što je pogodno za imena jer se mogu razlikovati po duljini.

10. (4 boda) Što ispisuje sljedeća SQL naredba nad zamišljenom tablicom "zaposlenici" s podacima osoba: `SELECT ime FROM zaposlenici WHERE prezime LIKE '%an' ORDER BY prezime DESC;`

ODGOVOR: SQL naredba SELECT ime FROM Zaposlenici WHERE prezime LIKE '%an' ORDER BY prezime DESC; ispisuje imena svih zaposlenika iz tablice Zaposlenici čija prezimena završavaju s 'an'. Rezultati su poredani prema prezimenima u silaznom (DESC) redoslijedu.

1. (1 bod) Što je Hibernate u odnosu na JPA (Java Persistence API)?

ODGOVOR: Hibernate je ORM okvir za Javu koji omogućuje mapiranje objekata na relacijske tablice i pruža dodatne značajke izvan JPA specifikacije. JPA je standardni API za ORM u Javi koji definira skup anotacija i API-ja za upravljanje perzistencijom objekata. Hibernate može biti korišten kao implementacija JPA standarda.

12. (4 boda) Ako su klase Person i Address anotirane na sljedeći način, napišite nazive tablica i njihovih stupaca koje će nastati u bazi podataka.

```
@Entity
@Table(name = "Address")
public class Address {

    @Id @GeneratedValue
    @Column(name = "id", unique = true, nullable = false)
    private int id;

    @Column(name = "street")
    private String street;

    @Column(name = "number")
    private int number;

    @Column(name = "place")
    private String place;

    @Column(name = "code")
    private String code;

    @Column(name = "country")
    private String country;

    ...
}

@Entity
@Table(name="Person")
public class Person {

    @Column(name="first_name")
    private String firstName;

    @Column(name="last_name")
    private String lastName;

    @Id
    @Column(name="id", unique = true, nullable = false)
    private String id;

    @ManyToOne(cascade=CascadeType.PERSIST)
    @JoinColumn(name="address_id")
    private Address address;

    ...
}
```

Tablica: Address

Stupci:
id (tip: integer, jedinstveni, obavezan)
street (tip: string)
number (tip: integer)
place (tip: string)
code (tip: string)
country (tip: string)
Tablica: Person

Stupci:
id (tip: string, jedinstveni, obavezan)
first_name (tip: string)
last_name (tip: string)
address_id (tip: integer, vanjski ključ koji upućuje na id stupac u tablici Address)