

Score: 1.000 (=100.0%)

Id: 45740

Kod izrade **skalabilnih arhitektura aplikacija** označite sve tvrdnje u vezi načela jednostavnosti **koje su istinite**:

470/598

Načela skalabilnosti

- ☒ **a** Jednostavnost se može postići korištenjem TTD (*test-driven development*) metodologije
- ☐ **b** Potrebno je **predvidjeti** svaki mogući scenarij i svaki rubni uvjet
- ☐ **c** Jednostavnost se **ne** može postići korištenjem TTD (*test-driven development*) metodologije, štoviše TTD je **potrebno** uvijek izbjegavati
- ☐ **d** Načelo jednostavnosti temelji se na korištenju jednostavnih tehnologija na poslužiteljima
- ☒ **e** Načelo jednostavnosti temelji se na načelima objektno orijentiranog programiranja i cilj je postići lokalnu jednostavnost strukture kôda
- ☒ **f** Potrebno je **izbjeci predviđanje** svakog mogućeg scenarija i svakog rubnog uvjeta jer se time gubi fokus s najčešćih scenarija

Score: -0.250 (~-25.0%)

Id: 45644

Ako su `Milk` i `Beverage` definirani s

```
type Milk = {  
  brand : string;  
  fat: number;  
  volume: number;  
}
```

```
type Beverage = {  
  type : string;  
  brand : string;  
  volume : number;  
}
```

što će ispisati sljedeći programski odsječak

```
let x : Milk | Beverage = {  
  type: 'juice',  
  brand : 'Home made',  
  volume : 2  
};  
console.log(typeof x);  
console.log(typeof (x as Beverage));
```

**a**

Milk | Beverage  
Milk | Beverage

**b**

object  
object

**c**

Beverage  
Beverage

**d**

Milk | Beverage  
Beverage

**e**

object  
Beverage

Score: -0.250 (-25.0%)

Id: 45664

Koje od sljedećih izjava su točne?

**a** Prilikom iscrtavanja odnosno osvježavanja prikaza, Vue koristi tzv. **Virtual DOM** kako bi optimirao taj proces.

**b** Vue ne omogućuje dvosmjerno povezivanje (*two way binding*)

**c** Vue programeru izlaže svoj životni ciklus (*lifecycle events*) na način da postoje funkcije (*lifecycle hooks*) **prije i poslije** svakog od tih događaja, npr. `onBeforeMount()` i `onMounted()`

**d** Na jednoj stranici može biti samo jedna Vue aplikacija

**e** Vue nema vlastiti ugrađeni *state management library*, već se najčešće koristi (dodatno instalira) vanjski library koji se zove Vuex.

**f** Vue ima vlastiti ugrađeni *state management library* koji se zove Vuex.

Score: 1.000 (=100.0%)

Id: 45622

Ako je u Typescriptu napisan sljedeći kod

```
function sum(a, b) {  
  return a + b;  
}
```

kojeg tipa je povratna vrijednosti iz funkcije sum?

**a**

any

**b**

object

**c**

number

**d**

function

**e**

never

Score: 0,000 (=0.0%)

Id: 45742

Kod izrade **skalabilnih arhitektura aplikacija** označite sve tvrdnje u vezi načela otvoreno-zatvoreno koje su istinite:

**a** Središnji cilj ovog načela je **potpuno onemogućiti** buduće izmjene softvera i time potpuno ukloniti **sve** troškove promjena

**b** Ovo načelo odnosi se na stvaranje kôda koji se **mora** mijenjati kada se promijene zahtjevi ili kada se pojave novi obrasci uporabe

**c** Kôd je **otvoren** za proširenje i **zatvoren** za izmjenu

**d** Kôd je **zatvoren** za proširenje i **otvoren** za izmjenu

**e** Ovo načelo odnosi se na stvaranje kôda koji se **ne** mora mijenjati kada se promijene zahtjevi ili kada se pojave novi obrasci uporabe

**f** Središnji cilj ovog načela je **povećati fleksibilnost** softvera i učiniti buduće promjene **jeftinijima**

Score: 1.000 (=100.0%)

Id: 45739

Kod izrade **skalabilnih arhitektura aplikacija** označite sve tvrdnje u vezi vertikalnog skaliranja koje su istinite:

**a**

Vertikalnim skaliranjem povećaju se performanse **postojećih** poslužitelja ili resursa

**b**

Vertikalnim skaliranjem dodaje se više **istih** poslužitelja ili resursa

**c**

Kod vertikalnog skaliranja **ne postoji** *scale down* već samo *scale up*

**d**

Kod vertikalnog skaliranja osim *scale up* postoji i *scale down* jer kvalitetan skalabilan sustav trebao bi omogućiti i smanjenje kapaciteta

**e**

Vertikalno skaliranje pogodno je za male web aplikacije gdje se kapacitet može održavati **povećanjem** kapaciteta postojećih poslužitelja i veličine već korištenih resursa

**f**

Vertikalno skaliranje **nije** pogodno za male web aplikacije gdje se kapacitet može održavati povećanjem kapaciteta postojećih poslužitelja i veličine već korištenih resursa

Score: -0.250 (= -25.0%)

Id: 45673

Koje od sljedećih tvrdnji vezane uz *service worker* **su istinite**?

**a**

Service worker se izvodi se mimo glavne preglednikove UI dretve.

**b**

Service worker radi samo ako je stranica poslužena putem HTTPS-a ili preko localhosta.

**c**

Service worker ne može koristiti *sinkroni* localStorage API, pa zato tipično koristi asinkroni IndexedDB.

**d**

Kod prvog otvaranja stranice, *service worker* moramo **registrati** nakon čega preglednik **dohvaća, instalira i aktivira** SW čime on preuzima kontrolu nad *klijentima* u svoj opsegu (*scope*).

**e**

Service worker nema pristup DOM-u.

**f**

Kada zatvorimo sve kartice/preglednike s domenom/scopem koje je neki service worker kontrolirao, onda se se gasi i prestaje raditi i taj service worker.

Score: 0.000 (=0.0%)

Id: 45674

Koje od sljedećih tvrdnji **su istinite**?

**a** Dozvolom za notifikaciju, web-aplikacija dobija i implicitnu dozvolu za push notifikacije te ju ne mora posebno tražiti.

**b** Push notifikacije uvelike ovise o preglednicima jer se oslanjaju na push service infrastrukturu u oblaku, koju svaki proizvođač preglednika implementira zasebno

**c** VAPID ključ nam treba kako bi od korisnika zatražili dozvolu za korištenje push notifikacija

**d** Notification API je nezavisan od service workera i može se koristiti i bez njega

**e** Web-aplikacija ne može prikazati notifikaciju bez eksplicitne dozvole korisnika,

**f** Push notifikacija neće biti primljena ako su ugašeni svi prozori preglednika



Score: 0.000 (=0.0%)

Id: 45643

Neka su `Milk`, `T` i `x` definirani kao u nastavku.

```
type Milk = {  
  brand : string;  
  fat: number;  
  volume: number;  
}
```

```
type T = keyof Milk;
```

Označite ispravne naredbe ako je u postavkama TypeScriptovog prevodioca opcija `strictNullChecks` postavljena na `false`.

**a** `let x : T = "brand" | "fat" | "volume";`

**b** `let x : T = "brand";`

**c** `let x : T;`

**d** `let x : T = "Milk";`

**e** `let x : T = 1;`

Score: 0.000 (=0.0%)

Id: 45626

Ako su `Milk` i `Beverage` definirani kao u nastavku, što je `Milk & Beverage`?

```
type Milk = {  
  brand : string;  
  fat: number;  
  volume: number;  
}
```

```
type Beverage = {  
  type : string;  
  brand : string;  
  volume : number;  
}
```

**a** tip koj sadrži sljedeće svojstva

```
{  
  brand : string;  
  fat: number;  
  type : string;  
  volume: number;  
}
```

**b** `any`

**c** tip koji u sebi sadrži

```
{  
  brand : string;  
  volume: number;  
}
```

**d** tip koj sadrži sljedeće svojstva

```
{  
  brand : string[];  
  fat: number;  
  type : string;  
  volume: number[];  
}
```

**e** tip koji dozvoljava da se u varijablu tog tipa pohrani objekt sa svojstvima identičnim kao u `Milk` ili objekt sa svojstvima identičnim kao u `Beverage`

Score: 0.250 (=25.0%)

Id: 45683

Prilikom **određivanja performansi web sjedišta** označite sve što je potrebno provesti:

**a**

Procjenu troškova *deploymenta*

**b**

Test opterećenja (*load test*)

**c**

Ispitivanje kapaciteta baze podataka (*volume testing*)

**d**

Ispitivanje responzivnosti (UX)

**e**

Ispitivanje izdržljivosti (*endurance testing*)

**f**

Procjenu sigurnosnih ranjivosti

Score: 0.000 (=0.0%)

Id: 45639

Ako je `Milk` definiran kao u nastavku, kako definirati `Beverage` koji za razliku od mlijeka nema `fat`, ali ima svojstvo `type` koje bi predstavljalo naziv vrste pića?

```
type Milk = {  
  brand : string;  
  fat: number;  
  volume: number;  
}
```

a

```
type Beverage = Milk - {"fat"} + {"type" : string};
```

b

```
type Beverage = Partial<Milk, "fat"> & "type";
```

c

```
type Beverage = {keyof Milk - {"fat"}} & {"type" : string};
```

d

```
type Beverage = Omit<Milk, "fat"> & {"type" : string};
```

e

```
type Beverage = Pick<Milk, "fat"> & {"type" : string};
```

Označite sve ispravne programske odsječke.

**a**

```
function pickRandomly(data : string) : string;
function pickRandomly<T>(data : string | T[]) : string | T {
  const pos = Math.floor(Math.random() * data.length);
  return data[pos];
}

let s = "Web2";
let c : string = pickRandomly(s);
console.log(c);
```

**b**

```
function pickRandomly<T>(data : T[]) : T {
  const pos = Math.floor(Math.random() * data.length);
  return data[pos];
}

function pickRandomly(data : string) : string {
  const pos = Math.floor(Math.random() * data.length);
  return data[pos];
}

let s = "Web2";
let c : string = pickRandomly(s);
let a = [10, 20, 30, 40];
let n : number = pickRandomly(a);
```

**c**

```
function pickRandomly<T>(data : T[]) : T;
function pickRandomly<T>(data : string | T[]) : string | T {
  const pos = Math.floor(Math.random() * data.length);
  return data[pos];
}

let a = [10, 20, 30, 40];
let n : number = pickRandomly(a);
console.log(n);
```

**d**

```
function pickRandomly(data : string) : string;
function pickRandomly<T>(data : T[]) : T;
function pickRandomly<T>(data : string | T[]) : string | T {
  const pos = Math.floor(Math.random() * data.length);
  return data[pos];
}

let s = "Web2";
let c : string = pickRandomly(s);
let a = [10, 20, 30, 40];
let n : number = pickRandomly(a);
```

**e**

```
function pickRandomly<T>(data : string | T[]) : string | T {
  const pos = Math.floor(Math.random() * data.length);
  return data[pos];
}

let s = "Web2";
let c : string = pickRandomly(s);
console.log(c);
```

Score: 1.000 (=100.0%)

Id: 45738

Kod izrade **skalabilnih arhitektura aplikacija** označite sve tvrdnje u vezi horizontalnog skaliranja **koje su istinite**:

**a** Kod horizontalnog skaliranja osim *scale out* postoji i *scale in* jer kvalitetan skalabilan sustav trebao bi omogućiti i smanjenje kapaciteta

**b** Horizontalnim skaliranjem povećaju se performanse **postojećih** strojeva ili resursa

**c** Horizontalnim skaliranjem dodaje se **više** fizičkih strojeva ili resursa

**d** Horizontalnim skaliranjem smanjuje se opterećenje na **svakom** stroju ili resursu

**e** Horizontalnim skaliranjem povećava se opterećenje na **svakom** stroju ili resursu

**f** Kod horizontalnog skaliranja **ne postoji** *scale in* već samo *scale out*

Score: 1.000 (=100.0%)

Id: 45624

Ako u argumentu funkcije izostavimo tip podatka, TypeScript će smatrati da je taj argument tipa *any*. Ako želimo da takav kod uzrokuje sintaksne pogreške prilikom prevođenja treba uključiti opciju

**a**

`noImplicitAny`

**b**

`resolveJsonModule`

**c**

`noEmitOnError`

**d**

`declaration`

**e**

`strictNullChecks`

Score: -0.250 (=-25.0%)

Id: 45641

Označite odgovor s ispravnim Typescript kodom koji bi omogućio da se sljedeći programski odsječak uspješno prevede i ispiše 5.

```
let p : ImmutablePoint = new ImmutablePoint(2, 3);  
console.log(p.x + p.y);
```

**a**

```
class ImmutablePoint {  
  readonly x, y;  
  constructor(x: number, y:number) { }  
}
```

**b**

```
class ImmutablePoint {  
  public readonly x : number;  
  public readonly y : number;  
  constructor(x: number, y:number) { }  
}
```

**c**

```
class ImmutablePoint {  
  public readonly x : number;  
  public readonly y : number;  
  constructor(public readonly x: number, public readonly y:number) {  
    this.x = x;  
    this.y = y;  
  }  
}
```

**d**

```
class ImmutablePoint {  
  constructor(public readonly x: number, public readonly y:number) {}  
}
```

**e**

```
interface ImmutablePoint {  
  constructor(readonly x: number, readonly y:number);  
}
```



Score: 0.000 (=0.0%)

Id: 45653

Što će se ispisati?

```
let obj = { broj: 8 };
let pObj = new Proxy(obj, {
  set: function (obj, prop, value) {
    if (prop === "broj") {
      if (value > 10) value = 10;
    }
    obj[prop] = value;
    return true; // Indicate success
  },
});
console.log(obj.broj);
console.log(pObj.broj);
obj.broj = 15;
console.log(obj.broj);
console.log(pObj.broj);
pObj.broj = 19;
console.log(obj.broj);
console.log(pObj.broj);
```

Student's answer:

```
8
8
15
8
10
19
```

Hint: 8 8 15 8 10 19 <> 8 8 15 15 10 10

Score: -0.250 (=-25.0%)

Id: 45662

Prednosti arhitekture u kojoj koristimo samo **jednostranične web-aplikacije** (u odnosu na druge dvije arhitekture) su:

a

Lagano testiranje

b

Jednostavnost

c

Sadržaj se lako prilagodi klijentu

d

Performanse (dio opterećenja se prebacuje na klijenta)

e

Responzivnost (UX)

f

Sigurnost

Score: -0.250 (=-25.0%)

Id: 45640

Neka je `Milk` definiran kao u nastavku.

```
type Milk = {  
  brand : string;  
  fat: number;  
  volume: number;  
}
```

Pretpostavimo da želimo napisati sljedeći kod

```
let milks : Milks = {};  
milks.DukatMali = {brand: "Dukat", volume: 0.5, fat : 3.2}  
milks.VindijskaVelika = {brand: "Vindijska", volume: 1.75, fat : 3.2}
```

Što od navedenog treba prethoditi takvom kodu?

**a** `interface Milks {  
 [key:Milk]  
};`

**b** `interface Milks {  
 [key:string] : Milk;  
};`

**c** `type Milks = {string : Milk}[];`

**d** `interface Milks {  
 any:Milk  
};`

**e** `type Milks = Milk[];`

Score: -0.250 (=-25.0%)

Id: 45642

Neka su u nekoj TypeScript datoteci zadani sučelje `Point` i funkcija `distance` kao u nastavku

```
interface Point {  
  x:number;  
  y:number;  
}  
  
function distance(p1:Point, p2: Point) : number {  
  return Math.sqrt((p1.x - p2.x) **2 + (p1.y - p2.y) ** 2);  
}
```

Označite sve programske odsječke koji su sintaksno ispravni.

**a**

```
let d = distance({x:1, y:2}, {x:4, y:6});
```

**b**

```
const point1 = new Point(1, 2);  
const point2 = new Point(4, 6);  
let d = distance(point1, point2)
```

**c**

```
const point1 : Point = {x: 1, y: 2};  
const point2 : Point = new Point(4, 6);  
let d = distance(point1, point2);
```

**d**

```
const vector = {x : 1, y: 2, z : 3};  
const point = {x : 2, y: 1};  
let d = distance(point, vector);
```

**e**

```
const vector = {x : 1, y: 2, z : 3};  
let d = distance(vector, vector);
```