Projektni zadatak PNRS1 – Zadatak 4

Android aplikacija „Pametna bolnica“

Ovogodišnji projektni zadatak biće izrada Android aplikacije pod nazivom **Smart Hospital**. Projekat se sastoji iz šest celina koje se nastavljaju jedna na drugu i zajedno čine kompletnu aplikaciju. Razvoj aplikacije se odvija u okruženju Android Studio, a rešenje je potrebno testirati na emulatoru.

Pred vama je tekst zadatka koji predstavlja četvrtu celinu. Ukupan broj bodova koji možete ostvariti na ovom zadatku je 10 bodova. Ovaj zadatak predstavlja nastavak na prethodne zadatke.

Zadatak 4 – HTTP

Razvoj zadatka nastaviti u već kreiranom projektu **Smart Hospital**. Ukoliko je potrebno, prethodni zadatak izmeniti i prilagoditi potrebama ovog zadatka.

HTTP Server

U prilogu zadatka nalazi se HTTP server koji je potrebno da svako za sebe pokrene lokalno.

Najpre je potrebno instalirati MongoDB. Uputsvo za instalaciju se nalazi u priloženom dokumentu „MongoDBInstall“. Sledeći koraci su:

Pozicionirajte se u direktorijum zadatak4/server i pokrenite sledeće komande:

Pokretanje servera izvodi se izdavanjem komande: **node index.js**

Kako bi se ova komanda mogla sprovesti, potrebno je prethodno preuzeti i instalirati **NODE**. Možete ga preuzeti na ovom linku (https://nodejs.org/en/download/) i ispratiti jednostavan proces instalacije.

Zajedno sa *node*, preuzeće se i ***npm*** *(node package manager*) pomoću kojeg možete preuzeti sve potrebne pakete.

Komande za preuzimanje paketa su:

```
npm install express
```

```
npm install mongoose
```

Nakon ovoga, pokrenite komandu **node index.js** i ukoliko nema grešaka, server je pokrenut. Nakon pokretanja, u konzoli će se ispisati poruka: “*Smart Hospital server is running on port 8080*“. Server možete zaustaviti komandom CTRL+C. ***Dok aplikacija upućuje zahteve ka serveru, on mora biti pokrenut!***

Ukoliko se i nakon preuzimanja paketa pojavljuje greške koje indikuju da nedostaje neki paket, probajte da ga preuzmete sa komandom npm instal ime_nedostajućeg_paketa.

HTTP Klijent

U okviru postojeće aplikacije, potrebno je implementirati HTTP klijenta koji komunicira sa Smart Hospital HTTP Serverom.

URL servera je: http://localhost:8080/api

Dobavljanje liste uređaja

Pre prikaza liste uređaja, potrebno je uputiti **HTTP GET** zahtev ka serveru na ruti: http://localhost:8080/api/devices/

Ovaj poziv kao odgovor vraća niz JSON objekata. Svaki uređaj predstavljen je kao JSON objekat koji se sastoji od *imena*, *id*-a (koji treba da bude jedinstven za svaki uređaj) i *stanja* uređaja (stanje uređaja može biti on/off i odgovara položaju switch-a tog elementa). Primer odgovora sa servera je prikazan u nastavku:

```
[
    {
        "name": "uredjaj1",
        "id": "5",
        "state": "on"
    },
    {
        "name": "uredjaj2",
        "id": "6",
        "state": "on"
    },
    {}
]
```

Iz ovog odgovora potrebno je preuzeti sve podatke, parsiranjem JSON-a. Ove podatke je potrebno čuvati u SQL bazi podataka koja je i do sada bila korišćena, te prikaz liste uređaja ostaje nepromenjen, tj. potrebno je sve informacije preuzimati i dalje iz baze, sa razlikom da se baza sada ne popunjava „ručno“ iz programa već parsiranjem odgovora od HTTP servera.

Popunjavanje liste uređaja

Lista uređaja je inicijalno prazna, te je potrebno popuniti. Ovo se radi upućivanjem **HTTP POST** zahteva ka ruti: http://localhost:8080/api/device

U okviru body-ja zahteva potrebno je proslediti JSON objekat sa detaljima uređaja (ime, id, stanje uređaja i tip – senzor ili aktuator). Primer tela zahteva dat je u nastavku:

```
{
        "name": "uredjaj3",
        "id": "7",
```

```
    "state": "off",
    "type": "sensor"
}
```

Odgovor od servera sadrži **poruku** i **kod**, u JSON objektu. Ovaj odgovor je potrebno protumačiti, te ispisati poruku greške ukoliko je do nje došlo, tj. ukoliko se kod statusa odgovora razlikuje od **200**. Primer uspešnog kreiranja uređaja dat je u nastavku:

```
{
    "message": "OK",
    "code": 200
}
```

Pribavljanje detalja uređaja

Prilikom pritiska na jedan uređaj otvara se novi *Activity* koji je do sada bio prazan. Sada je u ovom activity-ju potrebno prikazati naziv uređaja, njegov id, stanje uređaja i tip uređaja. Sve ove vrednosti prikazati kroz TextView komponente, a vrednosti dobaviti upućivanjem **HTTP GET** metode ka serveru, na ruti: http://localhost:8080/api/device/:id (*primer*: http://localhost:8080/api/device/5). Primer uspešnog odgovora dat je u nastavku:

```
{
    "_id": "60a2a490fccbc53100ff7a91",
    "name": "uredjaj1",
    "id": "5",
    "state": "off",
    "type": "actuator",
    "__v": 0
}
```

Obratiti pažnju da id koji se šalje u url-u zahteva, kao i onaj koji je potrebno prikazati je pod nazivom "id". Element sa ključem "_id" se vezuje za MongoDB bazu podataka koju server koristi i neće biti korišćen.

Promena stanja uređaja

Svaki put kada dođe do promene stanja uređaja (pomeranje switch-a), potrebno je ažurirati taj uređaj i na serveru pametne bolnice. To se postiže upućivanjem HTTP POST zahteva ka serveru, na ruti:

http://localhost:8080/api/device/:id/:state (*primer*: http://localhost:8080/api/device/5/off).

Ukoliko je uspešno izvršen zahtev, server će vratiti odgovor sa detaljima ovog uređaja i novim stanjem uređaja. Primer odgovora uspešnog zahteva dat je u nastavku:

```
{
    "_id": "60a2a490fccbc53100ff7a91",
    "name": "uredjaj1",
    "id": "5",
```

```
    "state": "off",
    "type": "actuator",
    "__v": 0
}
```


AddNewDeviceActivity

U **AdminView** prikazu dodati i dugme „Dodati uređaj“, iznad liste uređaja. Pritiskom na ovo dugme potrebno je otvoriti novi Activity, *AddNewDeviceActivity*. U okviru ovog prikaza treba implemenirati dve *EditText* komponente, za unos imena uređaja i za unos tipa uređaja, kao i dugme „Sačuvaj“. Pritiskom na ove dugme potrebno je uputiti zahtev ka HTTP serveru za popunjavanje liste uređaja, koji je prethodno opisan. Pri povratku na AdminView trebalo bi da uređaji budu vidljivi u listi.

Za potrebe zadatka dodati barem 5 različitih uređaja.

VAŽNE NAPOMENE:

1. Koristiti android.developers kao vid dokumentacije, pored materijala sa vežbi i predavanja.
2. Kako biste ostvarili maksimalan broj bodova, morate ispoštovati sve zahteve. Prostor za individualnost svakako ostaje na izboru boja i dimenzija teksta.
3. Koristiti relativne jedinice pri zadavanju dimenzija elemenata.
4. Sve boje i stringovi su resursi aplikacije, te je neophodno da budu definisani u *strings.xml* i *colors.xml* datotekama.
5. Voditi računa o formatiranju koda.
6. Nastaviti ovaj zadatak na prethodni, vodeći računa da su eventualne greške iz prethodnog zadatka ispravljene.
7. Rok za izradu **je nedelja, 23.05. u ponoć**. Rešenja učitati u okviru Assignment-a na Canvasu. U ponedeljak i utorak će biti organizovane elektronske odbrane domaćeg zadatka, u terminu vežbi.

SREĆNO!