**Redis Tema 1: "SnowTrack" - Praćenje kapaciteta na skijalištima**

**Opis funkcionalnosti:**

Web aplikacija omogućava korisnicima uvid u real-time podatke o stanju na skijalištima, uključujući:

* Broj slobodnih mesta na parkingu.
* Trenutno stanje staza (otvoreno/zatvoreno).
* ~~Informacije o vremenskim uslovima.~~

**Specifičnosti Redis-a:**

1. **Hash**:
   * Svako skijalište ima svoje ključeve sa informacijama o kapacitetu.
   * Primer:
   * "skijaliste:Kopaonik": {
   * "parking\_slobodna\_mesta": 25,
   * "staza\_otvorena": 15,
   * "staza\_zatvorena": 3
   * }
2. **Pub/Sub**:
   * Sistem šalje obaveštenja korisnicima kada se promeni stanje (npr. zatvaranje staze zbog lošeg vremena).
3. **Sorted Set**:
   * Rangiranje skijališta na osnovu poseta korisnika ili broja aktivnih staza.
   * Primer:
   * zadd "popularnost\_skijalista" 1200 "Kopaonik"
4. **TTL**:
   * ~~Keširani podaci o vremenskim uslovima sa automatskim brisanjem svakih 30 minuta.~~

**UI Funkcionalnosti:**

* Prikaz liste skijališta sa trenutnim podacima.
* Notifikacije o promenama (npr. obaveštenje o gužvama).
* Filter za pronalaženje skijališta sa najviše slobodnih parking mesta.

**Minimalni tehnički zahtevi:**

* Backend: Node.js ili Python (Redis klijent poput redis-py ili ioredis).
* Frontend: React za prikaz podataka u realnom vremenu.
* Redis server sa konfiguracijom za Pub/Sub.

Naravno! Ako spojiš **SkiPlanner** sa Redis funkcionalnostima, na front-endu bi se trebale dodati nove sekcije i opcije kako bi korisnici mogli da iskoriste te funkcionalnosti. Evo kako bi to moglo izgledati i koje korake treba da preduzmeš:

**1. Nova funkcionalnost na front-endu**

Redis bi se koristio za real-time podatke i personalizovana iskustva, pa bi se na front-end dodale sledeće sekcije:

**a. Real-time kapacitet skijališta**

* **Nova stranica ili sekcija**: "Stanje na skijalištima".
* Prikaz liste svih skijališta sa informacijama u realnom vremenu:
  + Broj slobodnih parking mesta.
  + Status staza (otvorene/zatvorene).
  + Popularnost skijališta (broj posetilaca).

**Primer dizajna (UI):**

Skijalište | Parking mesta | Otvorene staze | Zatvorene staze | Popularnost

-------------------------------------------------------------------------------

Kopaonik | 25 | 15 | 3 | ⭐⭐⭐⭐

Tara | 10 | 10 | 5 | ⭐⭐⭐

Stara Planina | 50 | 20 | 0 | ⭐⭐⭐⭐⭐

* Koristi **real-time osvežavanje** pomoću WebSocket-a ili periodičnog fetch-ovanja Redis podataka sa servera.

**b. Notifikacije**

* Dodaj funkcionalnost za **obaveštenja u realnom vremenu**:
  + Korisnik dobija poruku kada:
    - Staza na skijalištu postane zatvorena.
    - Parking postane popunjen.
    - Upozorenje o gužvi.
  + UI prikazuje obaveštenja u vidu toast poruka ili banera.

**Primer:**

🔔 Staza "Jaram" na Kopaoniku je zatvorena zbog loših vremenskih uslova.

🔔 Parking na Stara Planina skijalištu je popunjen.

**c. Filter i rangiranje skijališta**

* Omogući korisnicima da **filtriraju skijališta**:
  + Sa najviše slobodnih parking mesta.
  + Sa najviše otvorenih staza.
  + Po popularnosti.

**Primer UI elementa za filter:**

Filtriraj skijališta po:

[ ] Najviše slobodnih parking mesta

[ ] Najviše otvorenih staza

[ ] Najpopularnija

[Primeni filter]

**2. Način integracije Redis funkcionalnosti u postojeći SkiPlanner front-end**

1. **Dodavanje API ruta**:
   * Backend bi trebao imati dodatne API endpoint-e za Redis funkcionalnosti, npr.:
     + GET /realTimeStatus: Vraća real-time podatke o skijalištima.
     + GET /notifications: Vraća obaveštenja za korisnika.
     + POST /subscribeNotifications: Omogućava korisniku da se pretplati na obaveštenja.
2. **Dodavanje novih sekcija na front-end**:
   * Na stranici za pregled skijališta dodaj **tabelu za real-time podatke**.
   * Na početnu stranicu dodaj sekciju za popularna skijališta (npr. "Top 5 skijališta trenutno").
3. **Implementacija real-time obaveštenja**:
   * Koristi WebSocket za prikaz notifikacija kada Redis Pub/Sub primi poruku.
   * Na primer, kada se status staze promeni, Redis Pub/Sub šalje signal backend-u, koji prosleđuje obaveštenje korisniku na front-end.

**3. Tehnički koraci za front-end integraciju**

**a. Real-time podaci**

* Na front-endu koristi framework kao što je **React Query** za povlačenje podataka sa servera.
* Primer fetch logike:

import { useQuery } from 'react-query';

const fetchRealTimeStatus = async () => {

const response = await fetch('/api/realTimeStatus');

return response.json();

};

const RealTimeStatus = () => {

const { data, isLoading } = useQuery('realTimeStatus', fetchRealTimeStatus);

if (isLoading) return <div>Loading...</div>;

return (

<table>

<thead>

<tr>

<th>Skijalište</th>

<th>Parking mesta</th>

<th>Otvorene staze</th>

<th>Popularnost</th>

</tr>

</thead>

<tbody>

{data.map((ski) => (

<tr key={ski.id}>

<td>{ski.name}</td>

<td>{ski.parking}</td>

<td>{ski.openTrails}</td>

<td>{ski.popularity}</td>

</tr>

))}

</tbody>

</table>

);

};

**b. Notifikacije**

* Koristi WebSocket ili **Server-Sent Events (SSE)** za notifikacije.
* Primer korišćenja WebSocket-a:

const NotificationSystem = () => {

useEffect(() => {

const socket = new WebSocket('ws://localhost:8080/notifications');

socket.onmessage = (event) => {

alert(`🔔 ${event.data}`);

};

return () => socket.close();

}, []);

return null;

};

**4. Predlog UI arhitekture**

1. **Početna stranica**:
   * Sekcija sa "Najpopularnijim skijalištima".
   * Prikaz real-time statusa parkinga i staza.
2. **Stranica skijališta**:
   * Detaljni podaci o parkingu i stazama za odabrano skijalište.
   * Mogućnost prijave za obaveštenja.
3. **Filter i pretraga**:
   * Korisnik može filtrirati skijališta na osnovu real-time podataka.