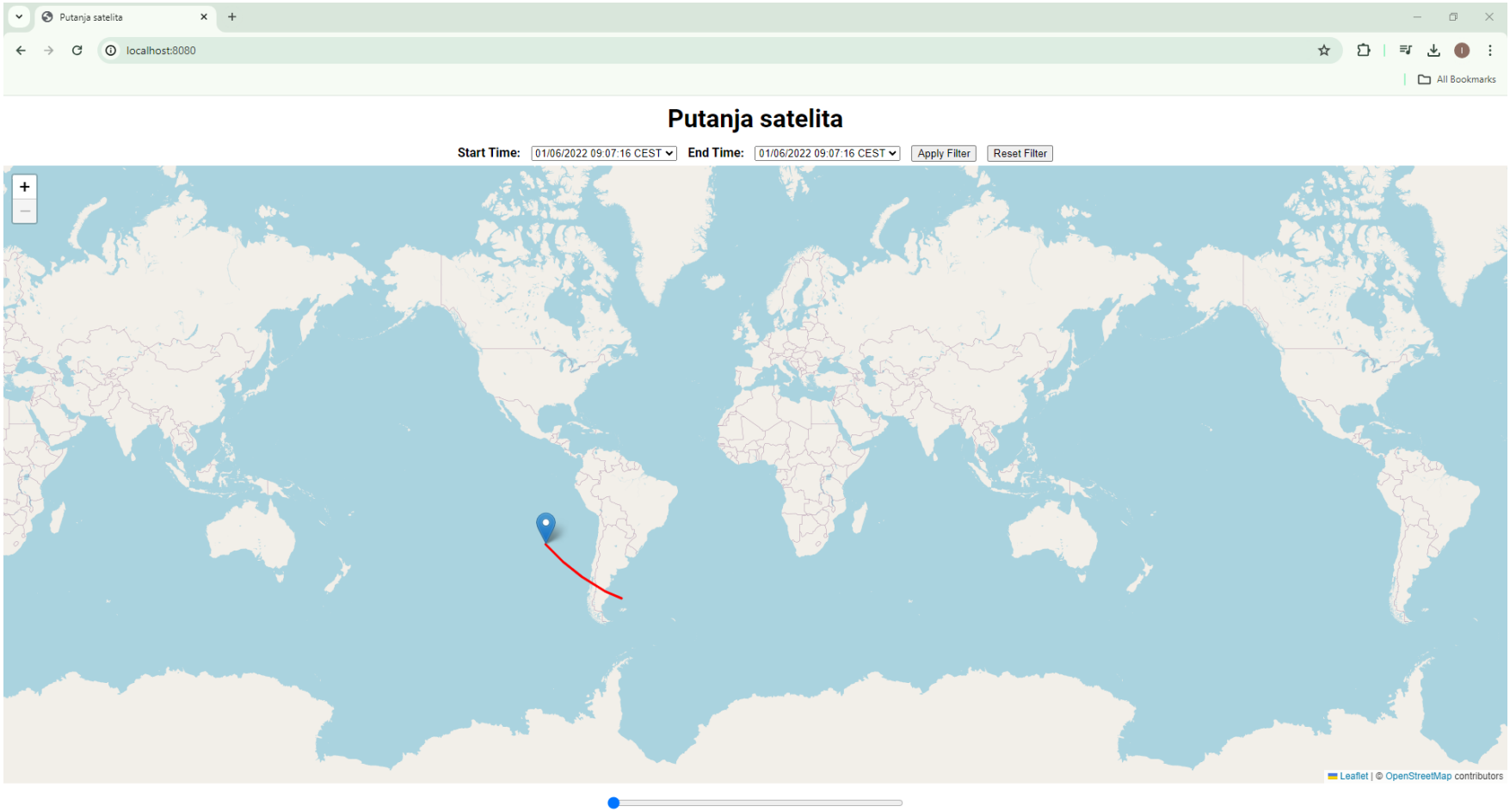
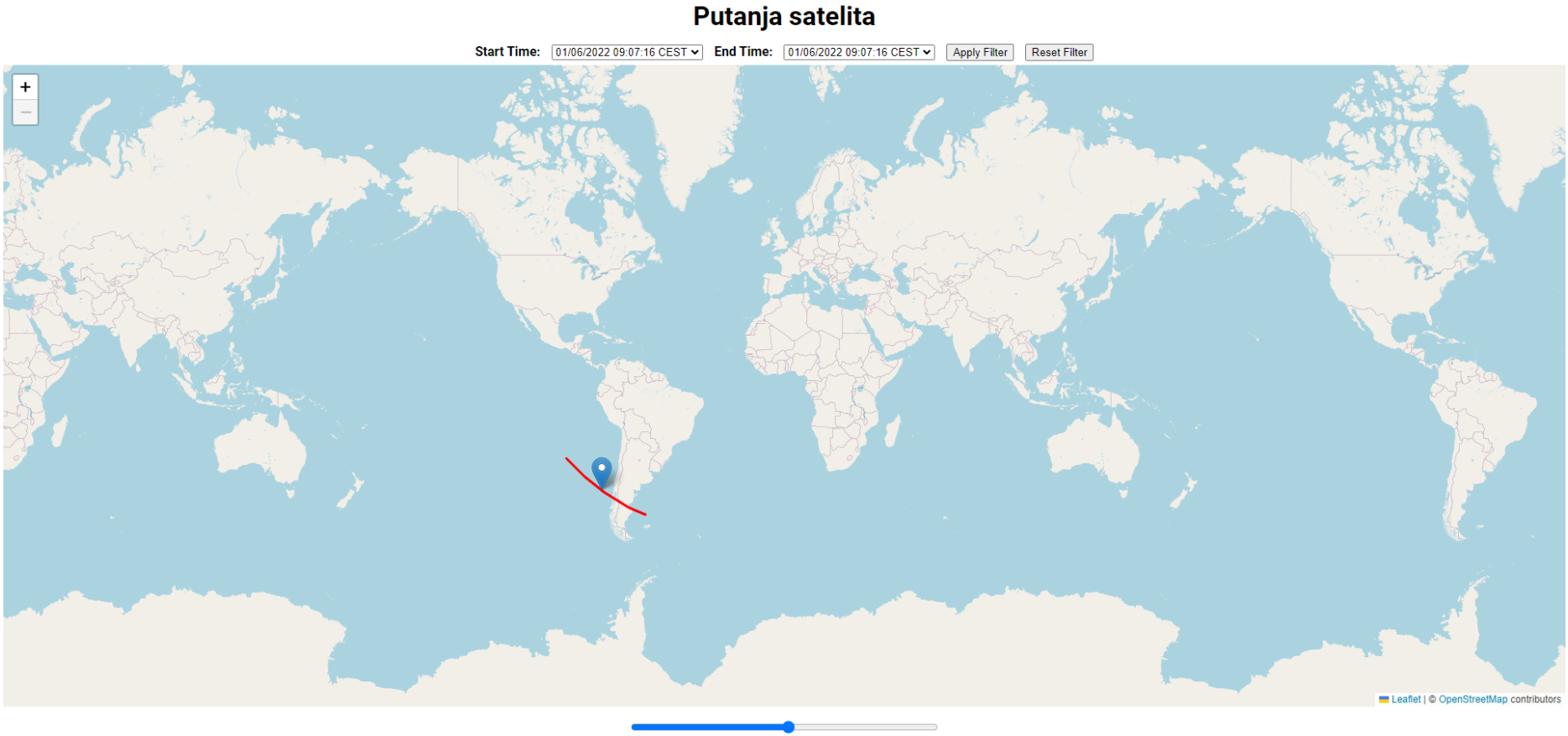
Pojašnjenje korisničkih mogućnosti na stranici:

Izgled stranice pri pokretanju koda u Command Prompt-u sljedećim naredbama:

1. javac WebServer.java
2. java WebServer



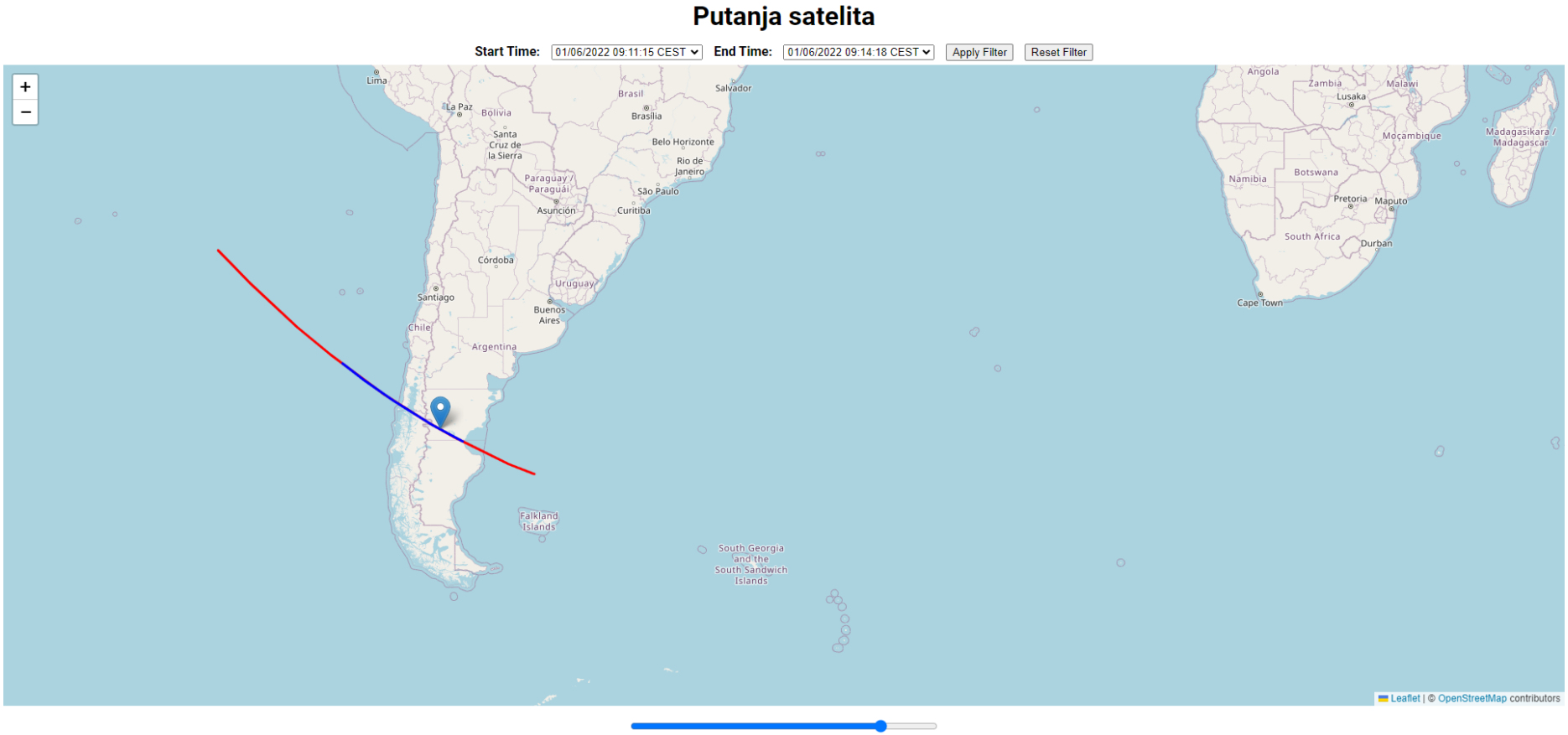
Pri pomicanju klizača (slider) ispod karte, pomičemo satelit po njegovoj putanji:



Pri filtriranju po vremenu što nam je opcija iznad karte i klikom na gumb *Apply Filter*, plavom se linijom na cijeloj putanji označi dio putanje koji smo odabrali i onda, također, možemo pomicati klizač ispod karte kako bismo pratili tu putanju:

Klikom na gumb Reset Filter, stanje na stranici vraća se u stanje koje bude pri samom pokretanju koda u Command Prompt-u.

Također, moguće je zumirati i „odzumirati“ kartu klikom na + i – u gornjem lijevom kutu karte:



Pojašnjenje dijelova koda:

1. index.html

HTML:

- Head dio:
  - ➔ Linkovi na vanjske stilove za Leaflet (popularna open-source JavaScript knjižnica za interaktivne karte) i Google Fonts
  - ➔ Postavljanje naslova stranice na „Putanja satelita"

- Body dio:
  - ➔ Sadrži naslov s tekstom „Putanja satelita"
  - ➔ *div* s *id*-om controls koji sadrži:
    1. Padajuće (dropdown) izbornike za odabir početnog i završnog vremena
    2. Gumbe za primjenu i resetiranje filtara
  - ➔ *div* s *id*-om *map* gdje se prikazuje karta
  - ➔ *div* s *id*-om *slider-controls* koji sadrži klizač (slider) za animiranje pozicije satelita
  - ➔ HTML kod u *body* dijelu:

```
<h1>Putanja satelita</h1>
<div id="controls">
    <label for="start-time">Start Time:</label>
    <select id="start-time" onchange="updateEndTimes()"></select>
    <label for="end-time">End Time:</label>
    <select id="end-time" onchange="updateStartTimes()"></select>
    <button onclick="applyFilter()">Apply Filter</button>
    <button onclick="resetFilter()">Reset Filter</button>
</div>
<div id="map"></div>
<div id="slider-controls">
    <input type="range" id="slider" min="0" max="100" value="0" oninput="moveSatellite()">
</div>
<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
```

CSS:

- Stilovi:
  - ➔ Tijelo stranice postavljeno je kao *flex container* s kolonijalnim smjerom, visinom od 100vh i skrivenim preljevanjem kako bi ispunilo ekran
  - ➔ *div* za kartu raste kako bi ispunio dostupni prostor.
  - ➔ Kontrole (padajući izbornici za odabir vremena, gumbi za primjenu i resetiranje te opcije i klizač) centrirane su i imaju malo paddinga
  - ➔ Klizač (slider) postavljen je na širinu od 10 cm

```
<style>
    body {
        margin: 0;
        padding: 0;
        display: flex;
        flex-direction: column;
        height: 100vh;
        overflow: hidden;
        font-family: 'Roboto', sans-serif;
    }

    #map {
        flex-grow: 1;
    }

    #controls, #slider-controls {
        text-align: center;
        padding: 5px;
        background: white;
    }

    #controls label {
        margin: 0 5px;
    }

    #controls select, #controls button {
        margin: 0 5px;
    }

    #slider-controls {
        margin-top: 10px;
    }

    #slider {
        width: 10cm;
    }

    h1 {
        text-align: center;
        font-family: 'Roboto', sans-serif;
        font-size: 2em;
        margin: 10px 0;
    }
</style>
```

JavaScript:

- Varijable i inicijalizacija:
  - ➔ *satellitePath*: Polje koje sadrži koordinate i vremenske oznake putanje satelita
  - ➔ *map*: Kreira kartu s Leaflet-om, postavlja granice i dodaje pločice (Tiles) OpenStreetMap-a
  - ➔ *satelliteLayer*, *polylineLayer*, *filteredLayer*: Slojevi za prikaz markera, polilinija i filtriranih podataka na karti

```
var satellitePath = [
    [-29.8998, -94.1934, 1654067236],
    [-30.148, -93.9219, 1654067241],
    // ...
    [-48.4835, -60.0146, 1654067751]
];

var map = L.map('map', {
    maxBounds: [[-90, -180], [90, 180]],
    maxBoundsViscosity: 1.0,
    minZoom: 2,
    maxZoom: 8
}).setView([0, 0], 2);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);

var satelliteLayer = L.layerGroup().addTo(map);
var polylineLayer = L.polyline([], { color: 'red' }).addTo(map);
var filteredLayer = L.layerGroup().addTo(map);
```

- Funkcija za ažuriranje karte – *updateMap(path)*: Ažurira kartu s novim putanjama:
  - ➔ Briše postojeće slojeve
  - ➔ Ako putanja sadrži točke, kreira poliliniju i markere za svaku točku
  - ➔ Dodaje marker za početnu točku putanje

```
function updateMap(path) {
    satelliteLayer.clearLayers();
    var latlngs = path.map(p => [p[0], p[1]]);
    polylineLayer.setLatLngs(latlngs);
    L.marker(latlngs[0]).addTo(satelliteLayer);
}
```

- Formatiranje datuma – *formatDate(timestamp)*: Formatira vremenske oznake u čitljiv oblik koristeći vremensku zonu Europe/Zagreb

```
function formatDate(timestamp) {
    var date = new Date(timestamp * 1000);
    var options = {
        year: 'numeric',
        month: '2-digit',
        day: '2-digit',
        hour: '2-digit',
        minute: '2-digit',
        second: '2-digit',
        timeZone: 'Europe/Zagreb',
        timeZoneName: 'short'
    };
    return new Intl.DateTimeFormat('en-GB', options).format(date).replace(',', '');
}
```

- Popunjavanje padajućih izbornika – *populateDropdowns()*: Popunjava početni i završni izbornik s vremenskim oznakama iz satellitePath
  - ➔ Briše postojeće opcije u izbornicima
  - ➔ Dodaje nove opcije iz *satellitePath*
  - ➔ Ažurira završne vremenske opcije

```
function populateDropdowns() {
    var startSelect = document.getElementById('start-time');
    var endSelect = document.getElementById('end-time');

    satellitePath.forEach(function(point) {
        var option = document.createElement('option');
        option.value = point[2];
        option.text = formatDate(point[2]);
        startSelect.add(option.cloneNode(true));
        endSelect.add(option);
    });

    startSelect.selectedIndex = 0;
    endSelect.selectedIndex = 0;
    startSelect.scrollTop = 0;
    endSelect.scrollTop = 0;
}
```

- Ažuriranje završnih vremena – *updateEndTimes()*: Ažurira dostupne opcije u završnom izborniku na temelju odabranog početnog vremena
  ➔ Pronalazi indeks odabranog početnog vremena
  ➔ Dodaje opcije za završna vremena koja dolaze nakon početnog vremena

```
function updateEndTimes() {
    var startTime = parseInt(document.getElementById('start-time').value);
    var endSelect = document.getElementById('end-time');
    var selectedEndTime = endSelect.value;

    endSelect.innerHTML = '';

    satellitePath.forEach(function(point) {
        if (point[2] >= startTime) {
            var option = document.createElement('option');
            option.value = point[2];
            option.text = formatDate(point[2]);
            endSelect.add(option);
        }
    });

    for (var i = 0; i < endSelect.options.length; i++) {
        if (endSelect.options[i].value == selectedEndTime) {
            endSelect.selectedIndex = i;
            break;
        }
    }

    endSelect.scrollTop = 0;
}
```

- Ažuriranje početnih vremena – *updateStartTimes()*: Ažurira dostupne opcije u početnom izborniku na temelju odabranog završnog vremena
  - ➔ Pronalazi indeks odabranog završnog vremena
  - ➔ Dodaje opcije za početna vremena koja dolaze prije završnog vremena

```
function updateStartTimes() {
    var endTime = parseInt(document.getElementById('end-time').value);
    var startSelect = document.getElementById('start-time');
    var selectedStartTime = startSelect.value;

    startSelect.innerHTML = '';

    satellitePath.forEach(function(point) {
        if (point[2] <= endTime) {
            var option = document.createElement('option');
            option.value = point[2];
            option.text = formatDate(point[2]);
            startSelect.add(option);
        }
    });

    for (var i = 0; i < startSelect.options.length; i++) {
        if (startSelect.options[i].value == selectedStartTime) {
            startSelect.selectedIndex = i;
            break;
        }
    }

    startSelect.scrollTop = 0;
}
```

- Primjena filtriranja – *applyFilter()*: Primjenjuje filtriranje na temelju odabranih vremena
  ➔ Filtrira *satellitePath* kako bi uključio samo podatke unutar odabranog raspona vremena
  ➔ Ažurira kartu s filtriranim podacima

```
function applyFilter() {
    var startTime = parseInt(document.getElementById('start-time').value);
    var endTime = parseInt(document.getElementById('end-time').value);

    if (startTime > endTime) {
        alert("Start time must be before end time.");
        return;
    }

    filteredPath = satellitePath.filter(function(point) {
        return point[2] >= startTime && point[2] <= endTime;
    });

    if (filteredPath.length === 0) {
        alert("No data available for the selected time range.");
        return;
    }

    filteredLayer.clearLayers();
    var latlngs = filteredPath.map(p => [p[0], p[1]]);
    var polylineSegment = L.polyline(latlngs, { color: 'blue' });
    filteredLayer.addLayer(polylineSegment);

    satelliteLayer.clearLayers();
    L.marker(latlngs[0]).addTo(satelliteLayer);

    document.getElementById('slider').max = filteredPath.length - 1;
    document.getElementById('slider').value = 0;
}
```

- Resetiranje filtriranja – *resetFilter()*: Resetira filtere i vraća izvorne podatke
  ➔ Ponovno popunjava izbornike
  ➔ Vraća početnu putanju na kartu

```
function resetFilter() {
    document.getElementById('start-time').innerHTML = '';
    document.getElementById('end-time').innerHTML = '';
    filteredLayer.clearLayers();
    filteredPath = satellitePath;
    updateMap(satellitePath);
    document.getElementById('slider').max = satellitePath.length - 1;
    document.getElementById('slider').value = 0;
    populateDropdowns();
}
```

- Pomicanje satelita – *moveSatellite()*: Pomicanje satelita na karti prema vrijednosti klizača
  ➔ Briše i postavlja marker na trenutnu poziciju na temelju klizača

```
function moveSatellite() {
    var slider = document.getElementById('slider');
    var index = slider.value;

    var latlng = (filteredPath.length > 0 ? filteredPath : satellitePath)[index];
    if (latlng) {
        satelliteLayer.clearLayers();
        L.marker([latlng[0], latlng[1]]).addTo(satelliteLayer);
    }
}
```

- Inicijalizacija kontrole klizača:
  - ➔ Postavlja minimalne i maksimalne vrijednosti za klizač
  - ➔ Poziva funkcije za popunjavanje izbornika i ažuriranje karte s početnim podacima

```
filteredPath = satellitePath;
updateMap(satellitePath);
document.getElementById('slider').max = satellitePath.length - 1;
document.getElementById('slider').value = 0;
populateDropdowns();
```

2. WebServer.java

- Importi:
  - ➔ *HttpServer*, *HttpHandler*, i *HttpExchange* iz *com.sun.net.httpserver* paketa za stvaranje i konfiguriranje HTTP servera
  - ➔ *IOException*, *OutputStream*, *InetSocketAddress*, *Files*, i *Paths* iz *java.io* i *java.net* paketa za rukovanje ulazno-izlaznim operacijama i putanjama datoteka

```
import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.HttpHandler;
import com.sun.net.httpserver.HttpExchange;

import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.file.Files;
import java.nio.file.Paths;
```

- *main* metoda:
  - ➔ Stvara server: Kreira HTTP server koji sluša na portu 8080
  - ➔ Dodaje kontekst: Postavlja korijenski kontekst ("/") koji će obrađivati zahtjeve pomoću razreda *MyHandler*
  - ➔ Postavlja izvršitelja: Koristi zadani izvršitelj (stvara novu nit za svaki zahtjev)
  - ➔ Pokreće server: Počinje rad servera i ispisuje poruku da je server pokrenut

```
public static void main(String[] args) throws IOException {
    HttpServer server = HttpServer.create(new InetSocketAddress(port:8080), backlog:0);
    server.createContext(path:"/", new MyHandler());
    server.setExecutor(executor:null);
    server.start();
    System.out.println(x:"Server started on port 8080");
}
```

- *handle* metoda (unutar razreda *MyHandler*):
  - ➔ Čitanje datoteke: Učitava sadržaj datoteke *index.html* u niz
  - ➔ Postavljanje zaglavlja odgovora: Postavlja *Content-Type* zaglavlje na *text/html; charset=UTF-8*
  - ➔ Slanje zaglavlja odgovora: Šalje HTTP status 200 OK s duljinom odgovora
  - ➔ Pisanje odgovora: Piše sadržaj datoteke *index.html* u tijelo odgovora i zatvara izlazni tok

```
static class MyHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange t) throws IOException {
        String response = new String(Files.readAllBytes(Paths.get(first:"index.html")));
        t.getResponseHeaders().set(key:"Content-Type", value:"text/html; charset=UTF-8");
        t.sendResponseHeaders(rCode:200, response.length());
        OutputStream os = t.getResponseBody();
        os.write(response.getBytes());
        os.close();
    }
}
```