<!doctype html>

<html lang="es">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width,initial-scale=1" />

<title>PayMe — Solicita tu ride</title>

<!-- Leaflet CSS (map) -->

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"

integrity="sha256-sA+4m2b5kz0P7v5Q2c2gT2EwQp4yHf3s6XoY7G6qzq0=" crossorigin=""/>

<!-- Font Awesome for icons -->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css"

integrity="sha512-Nz2e1s6zq3...(trimmed-for-integrity-check)...==" crossorigin="anonymous" referrerpolicy="no-referrer" />

<style>

:root{

--bg: #fbfdff;

--card: #ffffff;

--muted: #6b7280;

--primary: #2b8cff; /\* soft blue \*/

--accent: #00c48c; /\* green accent \*/

--glass: rgba(255,255,255,0.7);

--shadow: 0 6px 18px rgba(16,24,40,0.08);

font-family: Inter, ui-sans-serif, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial;

}

\*{box-sizing:border-box}

body{

margin:0;

background: linear-gradient(180deg,#fbfdff 0%, #f7fbff 100%);

color:#0f172a;

-webkit-font-smoothing:antialiased;

-moz-osx-font-smoothing:grayscale;

}

header{

display:flex;

align-items:center;

justify-content:space-between;

gap:1rem;

padding:18px 28px;

max-width:1200px;

margin:18px auto;

}

.brand{

display:flex;align-items:center;gap:14px;

}

.logo{

width:48px;height:48px;border-radius:10px;background:linear-gradient(135deg,var(--primary),#5ad0ff);

display:flex;align-items:center;justify-content:center;color:white;font-weight:700;box-shadow:var(--shadow);

font-size:20px;

}

.brand h1{margin:0;font-size:18px}

.brand p{margin:0;font-size:12px;color:var(--muted)}

.auth{

display:flex;gap:10px;align-items:center;

}

.btn{

background:var(--primary);color:white;padding:10px 14px;border-radius:10px;border:0;font-weight:600;cursor:pointer;

box-shadow: 0 6px 14px rgba(43,140,255,0.18);

}

.btn.secondary{

background:transparent;color:var(--primary);border:1px solid rgba(43,140,255,0.12);

box-shadow:none;font-weight:600;

}

main{

max-width:1200px;margin:0 auto;padding:8px 20px 60px;

display:grid;grid-template-columns:440px 1fr;gap:22px;align-items:start;

}

.card{

background:var(--card);border-radius:14px;padding:18px;box-shadow:var(--shadow);

}

/\* Left panel \*/

.hero{

padding:22px;background:linear-gradient(180deg,rgba(255,255,255,0.6),rgba(250,252,255,0.8));

border-radius:14px;

}

.hero h2{margin:0 0 8px;font-size:22px}

.hero p{margin:0 0 16px;color:var(--muted)}

.form-row{display:flex;gap:10px;margin-bottom:12px}

label{font-size:12px;color:var(--muted);display:block;margin-bottom:6px}

input[type="text"], input[type="email"], select{

width:100%;padding:12px;border-radius:10px;border:1px solid #e6eefc;background:#fbfdff;

outline:none;font-size:14px;

}

.estimate{

display:flex;align-items:center;gap:10px;padding:12px;border-radius:10px;background:linear-gradient(90deg, #f6fbff, #ffffff);

margin:8px 0;

}

.drivers-list{max-height:160px;overflow:auto;margin-top:8px}

.driver-item{display:flex;align-items:center;gap:12px;padding:8px;border-radius:10px}

.avatar{width:44px;height:44px;background:#eef6ff;border-radius:8px;display:flex;align-items:center;justify-content:center;font-weight:700;color:var(--primary)}

.muted{color:var(--muted);font-size:13px}

/\* Map \*/

#map { height: 74vh; border-radius:14px; }

/\* Ride status \*/

.status{

margin-top:12px;padding:12px;border-radius:10px;background:linear-gradient(180deg,#fff,#fbfcff);

}

.status strong{display:block;margin-bottom:6px}

footer{max-width:1200px;margin:28px auto;color:var(--muted);font-size:13px;text-align:center;padding-bottom:60px}

/\* Responsive \*/

@media (max-width:980px){

main{grid-template-columns:1fr; padding:8px 14px}

#map{height:56vh}

}

/\* small helpers \*/

.small{font-size:13px;color:var(--muted)}

.pill{padding:6px 10px;border-radius:999px;background:rgba(0,0,0,0.04);font-weight:600}

</style>

</head>

<body>

<header>

<div class="brand">

<div class="logo">PM</div>

<div>

<h1>PayMe</h1>

<p>Solicita y encuentra tu ride — rápido y sencillo</p>

</div>

</div>

<div class="auth">

<div class="small">¿Eres conductor? <a href="#" id="switchDriver">Entrar</a></div>

<button class="btn" id="openSignIn"><i class="fa fa-user"></i> Iniciar</button>

</div>

</header>

<main>

<!-- Left panel -->

<div class="card hero">

<h2>Solicita tu ride</h2>

<p class="muted">Escribe dónde estás y hacia dónde vas. Verás conductores cercanos en el mapa.</p>

<div style="margin-top:10px">

<div class="form-row">

<div style="flex:1">

<label for="name">Tu nombre</label>

<input id="name" type="text" placeholder="Ej: Ana Pérez" value="">

</div>

</div>

<div class="form-row">

<div style="flex:1">

<label for="pickup">Origen (lat,lng o dirección aproximada)</label>

<input id="pickup" type="text" placeholder="Ej: 40.4168,-3.7038 o 'Centro'">

</div>

</div>

<div class="form-row">

<div style="flex:1">

<label for="destination">Destino (lat,lng o dirección)</label>

<input id="destination" type="text" placeholder="Ej: 40.4300,-3.7000 o 'Aeropuerto'">

</div>

</div>

<div style="display:flex;gap:8px;margin-top:8px">

<button class="btn" id="requestRide"><i class="fa fa-paper-plane"></i> Solicitar ride</button>

<button class="btn secondary" id="locateBtn"><i class="fa fa-location-crosshairs"></i> Usar mi ubicación</button>

</div>

<div class="estimate" style="margin-top:14px">

<div style="flex:1">

<div class="small">Tarifa estimada</div>

<div id="fare" style="font-weight:700;font-size:18px">—</div>

<div class="muted" id="distanceInfo">—</div>

</div>

<div style="text-align:right">

<div class="small">Vehículo</div>

<div class="pill">Estándar</div>

</div>

</div>

<div style="margin-top:10px">

<div class="small">Conductores cercanos</div>

<div class="drivers-list" id="driversList"></div>

</div>

<div class="status" id="rideStatus">

<strong>Estado del viaje</strong>

<div id="statusText" class="muted">No hay viajes activos</div>

</div>

</div>

<div style="margin-top:14px;font-size:13px;color:var(--muted)">

<strong>Nota:</strong> Esta demo simula conductores y movimiento en el mapa. Para producción, conecta las operaciones a tu backend (API para búsquedas, asignación, pagos y WebSockets para ubicaciones en tiempo real).

</div>

</div>

<!-- Map -->

<div class="card">

<div id="map"></div>

</div>

</main>

<footer>

PayMe — Demo frontend • Para integrar pagos, pasarelas y autenticación, necesitas backend seguro. © <span id="year"></span>

</footer>

<!-- Leaflet JS -->

<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"

integrity="sha256-o9N1jGgLr0... (trimmed) ..." crossorigin=""></script>

<script>

// ====== UTILIDADES ======

const toRad = deg => deg \* Math.PI/180;

function haversine(lat1, lon1, lat2, lon2){

const R = 6371; // km

const dLat = toRad(lat2-lat1);

const dLon = toRad(lon2-lon1);

const a = Math.sin(dLat/2)\*\*2 + Math.cos(toRad(lat1))\*Math.cos(toRad(lat2))\*Math.sin(dLon/2)\*\*2;

const c = 2\*Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

return R\*c;

}

// Fare model: base + per\_km

function estimateFare(km){

const base = 1.50; // currency units

const perKm = 0.80;

const fare = base + perKm \* km;

return Math.max(fare, 3).toFixed(2);

}

// Parse lat,lng from "lat,lng" or fake address -> geocode stub (random near center)

function parseLocation(text){

text = (text || "").trim();

// If looks like coords:

const coordMatch = text.match(/^\\s\*([+-]?\\d+\\.?\\d\*)\\s\*,\\s\*([+-]?\\d+\\.?\\d\*)\\s\*$/);

if(coordMatch){

return { lat: parseFloat(coordMatch[1]), lng: parseFloat(coordMatch[2]), label: text };

}

// Common words -> fallback simulated positions near a default city center

// Default: Madrid-ish center (40.4168, -3.7038) to show map working for many users.

// In production, integrate with a geocoding API (Mapbox, Nominatim, Google).

const base = { lat: 40.4168, lng: -3.7038 };

// create deterministic pseudo-random offset based on text

let seed = 0;

for(let i=0;i<text.length;i++) seed = (seed\*31 + text.charCodeAt(i))%1000;

const offsetLat = ((seed % 40) - 20)/10000; // small offset

const offsetLng = (((seed\*7) % 40) - 20)/10000;

return { lat: base.lat + offsetLat, lng: base.lng + offsetLng, label: text || 'Ubicación aproximada' };

}

// ====== MAP SETUP ======

const map = L.map('map', { zoomControl: true }).setView([40.4168, -3.7038], 13);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

maxZoom: 19,

attribution: '&copy; OpenStreetMap'

}).addTo(map);

document.getElementById('year').innerText = new Date().getFullYear();

// Markers & state

let pickupMarker = null;

let destMarker = null;

let driverMarkers = {}; // id -> marker

let drivers = [];

let ride = null; // {id, riderName, pickup, dest, driverId, status}

// Create some simulated drivers near the center

function seedDrivers(){

const center = { lat:40.4168, lng:-3.7038 };

drivers = [];

for(let i=0;i<8;i++){

const d = {

id: 'drv\_' + (1000 + i),

name: ['Carlos','María','Jorge','Lucía','Sofía','Miguel','Rosa','Pedro'][i%8],

car: ['Toyota','Hyundai','Seat','Renault'][i%4],

rating: (4 + Math.round(Math.random()\*10)/10).toFixed(1),

lat: center.lat + (Math.random()\*0.06 - 0.03),

lng: center.lng + (Math.random()\*0.06 - 0.03),

busy: false

};

drivers.push(d);

}

}

seedDrivers();

function renderDrivers(){

const list = document.getElementById('driversList');

list.innerHTML = '';

drivers.forEach(d=>{

const el = document.createElement('div');

el.className = 'driver-item';

el.innerHTML = `

<div class="avatar">${d.name.charAt(0)}</div>

<div style="flex:1">

<div style="font-weight:700">${d.name} <span class="small muted">· ${d.car}</span></div>

<div class="muted">Rating ${d.rating} · ${d.busy?'<span style="color:#ff914d">Ocupado</span>':'Disponible'}</div>

</div>

<div style="text-align:right">

<div class="small">~${(getDistanceToCenter(d)).toFixed(1)} km</div>

<button class="btn secondary small" onclick="centerOnDriver('${d.id}')">Ubicar</button>

</div>

`;

list.appendChild(el);

// Map marker

if(!driverMarkers[d.id]){

const m = L.marker([d.lat, d.lng], { title: d.name, riseOnHover:true }).addTo(map);

m.bindPopup(`<strong>${d.name}</strong><br>${d.car}<br>Rating ${d.rating}`);

driverMarkers[d.id]=m;

} else {

driverMarkers[d.id].setLatLng([d.lat, d.lng]);

}

});

}

function getDistanceToCenter(d){

return haversine(40.4168, -3.7038, d.lat, d.lng);

}

window.centerOnDriver = function(id){

const d = drivers.find(x=>x.id===id);

if(d) map.setView([d.lat, d.lng], 15);

if(driverMarkers[id]) driverMarkers[id].openPopup();

}

renderDrivers();

// Periodically move drivers a tiny random bit to simulate activity

setInterval(()=> {

drivers.forEach(d=>{

if(!d.busy){

d.lat += (Math.random()-0.5)\*0.002;

d.lng += (Math.random()-0.5)\*0.002;

}

});

renderDrivers();

}, 3000);

// ====== FORMS & INTERACTIONS ======

const pickupInput = document.getElementById('pickup');

const destInput = document.getElementById('destination');

const fareEl = document.getElementById('fare');

const distanceInfo = document.getElementById('distanceInfo');

const statusText = document.getElementById('statusText');

const requestBtn = document.getElementById('requestRide');

const locateBtn = document.getElementById('locateBtn');

locateBtn.addEventListener('click',()=> {

if(navigator.geolocation){

navigator.geolocation.getCurrentPosition(pos=>{

const lat = pos.coords.latitude;

const lng = pos.coords.longitude;

pickupInput.value = lat.toFixed(6) + ',' + lng.toFixed(6);

map.setView([lat,lng], 14);

updateEstimate();

}, err=>{

alert('No se pudo obtener ubicación: '+err.message);

});

} else alert('Geolocalización no disponible en este navegador');

});

// When either pickup or dest changes, update estimate

pickupInput.addEventListener('input', updateEstimate);

destInput.addEventListener('input', updateEstimate);

function updateEstimate(){

const p = parseLocation(pickupInput.value);

const d = parseLocation(destInput.value);

const km = haversine(p.lat, p.lng, d.lat, d.lng);

distanceInfo.innerText = km>0 ? `${km.toFixed(2)} km` : '—';

fareEl.innerText = km>0 ? estimateFare(km) + ' €' : '—';

}

// Click on map to set pickup/destination (shift-click to set dest)

map.on('click', function(e){

const lat = e.latlng.lat.toFixed(6), lng = e.latlng.lng.toFixed(6);

if(e.originalEvent && e.originalEvent.shiftKey){

destInput.value = lat + ',' + lng;

} else {

pickupInput.value = lat + ',' + lng;

}

updateEstimate();

});

// REQUEST RIDE flow (simulated)

requestBtn.addEventListener('click',()=>{

const name = document.getElementById('name').value.trim() || 'Usuario';

const pickup = parseLocation(pickupInput.value);

const dest = parseLocation(destInput.value);

// Validation

if(!pickup || !dest){ alert('Por favor completa origen y destino.'); return; }

// Place markers

if(pickupMarker) map.removeLayer(pickupMarker);

if(destMarker) map.removeLayer(destMarker);

pickupMarker = L.marker([pickup.lat, pickup.lng], {icon: L.icon({iconUrl:'', iconSize:[0,0]})}).addTo(map)

.bindPopup(`<strong>Origen</strong><br>${pickup.label}`).openPopup();

destMarker = L.marker([dest.lat, dest.lng]).addTo(map).bindPopup(`<strong>Destino</strong><br>${dest.label}`);

// Estimate

const km = haversine(pickup.lat, pickup.lng, dest.lat, dest.lng);

fareEl.innerText = estimateFare(km) + ' €';

distanceInfo.innerText = `${km.toFixed(2)} km`;

// Create ride object

ride = {

id: 'ride\_' + Date.now(),

riderName: name,

pickup, dest,

driverId: null,

status: 'searching' // searching -> assigned -> en\_route -> completed

};

statusText.innerText = 'Buscando conductor disponible...';

simulateAssignDriver();

});

// Find nearest available driver and assign (simulated)

function simulateAssignDriver(){

// find closest driver that is not busy

let available = drivers.filter(d=>!d.busy);

if(available.length===0){

statusText.innerText = 'No hay conductores disponibles. Intenta de nuevo más tarde.';

return;

}

available.sort((a,b)=>{

const da = haversine(a.lat,a.lng, ride.pickup.lat, ride.pickup.lng);

const db = haversine(b.lat,b.lng, ride.pickup.lat, ride.pickup.lng);

return da-db;

});

const driver = available[0];

driver.busy = true;

ride.driverId = driver.id;

ride.status = 'assigned';

statusText.innerHTML = `Conductor asignado: <strong>${driver.name}</strong> — ${driver.car}. En camino a recogerte.`;

// flash/open popup on driver marker

if(driverMarkers[driver.id]) driverMarkers[driver.id].openPopup();

// simulate movement: driver goes to pickup, then to destination

simulateDriverMovement(driver);

}

// Move driver in small steps towards target

function simulateDriverMovement(driver){

let phase = 'toPickup'; // toPickup -> toDest -> done

const speedKmh = 30 + Math.random()\*25; // variable speed

const stepInterval = 1000; // ms

const stepKm = (speedKmh/3600) \* (stepInterval/1000); // km per interval

const interval = setInterval(()=>{

const target = (phase==='toPickup') ? ride.pickup : ride.dest;

const curLat = driver.lat, curLng = driver.lng;

const distKm = haversine(curLat, curLng, target.lat, target.lng);

if(distKm < 0.05){ // within ~50m

if(phase==='toPickup'){

phase = 'toDest';

ride.status = 'en\_route';

statusText.innerHTML = `Recogido por <strong>${driver.name}</strong>. En camino al destino.`;

} else {

// completed

clearInterval(interval);

ride.status = 'completed';

statusText.innerHTML = `Viaje completado. Gracias por usar <strong>PayMe</strong>!`;

driver.busy = false;

// driver remains near dropoff

driver.lat = target.lat + (Math.random()-0.5)\*0.002;

driver.lng = target.lng + (Math.random()-0.5)\*0.002;

renderDrivers();

return;

}

} else {

// move fractionally towards target

const frac = Math.min(0.999, stepKm / distKm);

const newLat = curLat + (target.lat - curLat) \* frac;

const newLng = curLng + (target.lng - curLng) \* frac;

driver.lat = newLat;

driver.lng = newLng;

if(driverMarkers[driver.id]){

driverMarkers[driver.id].setLatLng([driver.lat, driver.lng]);

// update popup position if open

}

// update status with ETA

const etaMin = Math.max(1, Math.round((distKm / (speedKmh))\*60));

if(phase==='toPickup'){

statusText.innerHTML = `Conductor <strong>${driver.name}</strong> llega en ~${etaMin} min.`;

} else {

statusText.innerHTML = `En camino: ~${etaMin} min al destino.`;

}

}

}, stepInterval);

}

// Seed with initial estimate

updateEstimate();

// Sign-in (client-only demo)

document.getElementById('openSignIn').addEventListener('click',()=>{

const name = prompt('Nombre para demo (será tu nombre en la sesión):','Usuario');

if(name) document.getElementById('name').value = name;

});

// SWITCH to driver mode (demo)

document.getElementById('switchDriver').addEventListener('click', (e)=>{

e.preventDefault();

alert('Modo conductor (demo): los conductores están simulados. Para activar un driver real, conecta el backend y la app del conductor.');

map.setView([40.4168, -3.7038], 13);

});

// Accessibility: handle Enter to request

document.addEventListener('keydown', (e)=>{

if(e.key==='Enter' && (document.activeElement === destInput || document.activeElement === pickupInput)){

requestBtn.click();

}

});

// Expose some state for debugging in console

window.\_\_PAYME\_DEMO = { drivers, ride, map };

</script>

</body>

</html>