NailMeet – Web App (Frontend + Backend)

Proyecto listo para subir **fácil y funcional**: frontend (React + Vite) + backend (Node.js + Express + Mercado Pago). Incluye Google Maps, flujo de reserva y pago.

🧱Estructura de carpetas

```
nailmeet/
├─ frontend/
│   ├─ index.html
│   ├─ vite.config.js
│   ├─ package.json
│   ├─ .env.example
│   └─ src/
│       ├─ main.jsx
│       ├─ App.jsx
│       ├─ Success.jsx
│       ├─ data.js
│       └─ styles.css
└─ backend/
    ├─ package.json
    ├─ server.js
    └─ .env.example
```

🗜️Frontend (React + Vite)

`frontend/package.json`

```
{
  "name": "nailmeet-frontend",
  "private": true,
  "version": "1.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview --port 5173"
  },
  "dependencies": {
    "@react-google-maps/api": "^2.20.1",
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.3.1",
    "vite": "^5.4.0"
  }
}
```

`frontend/vite.config.js`

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
})
```

`frontend/.env.example`

```
VITE_GOOGLE_MAPS_API_KEY=TU_API_KEY_DE_GOOGLE
VITE_MP_PUBLIC_KEY=TEST-xxxxxxxxxxxxxxxxxxxxxxxx
VITE_API_BASE_URL=http://localhost:3001
VITE_FRONTEND_URL=http://localhost:5173
```

Copiá este archivo a `.env` y completá los valores.

`frontend/index.html`

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>NailMeet</title>
    <link rel="icon" href="data:," />
    <!-- SDK Mercado Pago -->
    <script src="https://sdk.mercadopago.com/js/v2"></script>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

`frontend/src/data.js`

```
export const SERVICES = [
  { id: 'basica', name: 'Manicuría básica', duration: 45, price: 7000 },
  { id: 'semipermanente', name: 'Esmaltado semipermanente', duration: 60,
price: 11000 },
  { id: 'kapping', name: 'Kapping en gel', duration: 90, price: 17000 },
  { id: 'esculpidas', name: 'Uñas esculpidas', duration: 120, price: 25000 },
  { id: 'retirado', name: 'Retirado de gel/semip', duration: 30, price:
4000 },
]

export const SALONS = [
  {
    id: 'glow', name: 'Glow Nails Studio', rating: 4.8, reviews: 189,
distanceKm: 1.2, openNow: true,
    phone: '+54 9 11 5555-1111', whatsapp: '+54 9 11 5555-1111', address:
'Av. Cabildo 1234, CABA',
    lat: -34.5616, lng: -58.4516, services:
['basica','semipermanente','kapping','retirado'],
    images: [
      'https://images.unsplash.com/photo-1596462502278-27bfdc403348?
q=80&w=1400&auto=format&fit=crop',
      'https://images.unsplash.com/photo-1522335789203-aabd1fc54bc9?
q=80&w=1400&auto=format&fit=crop',
    ],
    pros: ['Higiene impecable','Atención puntual','Amplia variedad de
colores']
  },
  {
    id: 'nailbar', name: 'The Nail Bar', rating: 4.6, reviews: 98,
distanceKm: 2.4, openNow: true,
    phone: '+54 9 11 4444-2222', whatsapp: '+54 9 11 4444-2222', address:
'Scalabrini Ortiz 650, CABA',
    lat: -34.5876, lng: -58.4259, services:
['basica','semipermanente','esculpidas','retirado'],
    images: [
      'https://images.unsplash.com/photo-1544947950-fa07a98d237f?
q=80&w=1400&auto=format&fit=crop',
      'https://images.unsplash.com/photo-1519014816548-bf5fe059798b?
q=80&w=1400&auto=format&fit=crop',
    ],
    pros: ['Excelente relación precio-calidad','Buena música','Profesionales
con experiencia']
  },
  {
    id: 'mani', name: 'Mani & Co.', rating: 4.9, reviews: 312, distanceKm:
0.9, openNow: false,
    phone: '+54 9 11 3333-7777', whatsapp: '+54 9 11 3333-7777', address:
'Av. Rivadavia 7600, CABA',
    lat: -34.6262, lng: -58.4656, services:
['basica','kapping','esculpidas','retirado'],
    images: [
      'https://images.unsplash.com/photo-1582092728100-82331d2e3c88?
q=80&w=1400&auto=format&fit=crop',
      'https://images.unsplash.com/photo-1521590832167-7bcbfaa6381f?
q=80&w=1400&auto=format&fit=crop',
    ],
    pros: ['Diseños artísticos','Butacas cómodas','Tés y bebidas de
cortesía']
  },
]
```

frontend/src/styles.css

```
:root { --radius: 16px; }
* { box-sizing: border-box; }
body { margin:0; font-family: ui-sans-serif, system-ui, -apple-system, Segoe
UI, Roboto, Helvetica, Arial; color:#111; }
.container { max-width: 780px; margin: 0 auto; padding: 12px; }
.header { position: sticky; top:0; background: rgba(255,255,255,.8);
backdrop-filter: blur(8px); border-bottom: 1px solid #eee; padding: 10px
12px; display:flex; align-items:center; justify-content:space-between; z-
index:10 }
.h1 { font-size: 18px; font-weight: 700; }
.row { display:flex; gap: 8px; }
.input { flex:1; padding: 10px 12px; border:1px solid #ddd; border-radius:
999px; }
.btn { padding: 10px 14px; border:1px solid #ddd; border-radius: 999px;
background:#fff; cursor:pointer }
.btn.primary { background:#111; color:#fff; border-color:#111 }
.badge { padding: 4px 10px; border:1px solid #ddd; border-radius: 999px;
font-size:12px; }
.card { border:1px solid #eee; border-radius: var(--radius); padding: 12px;
box-shadow: 0 1px 2px rgba(0,0,0,.04); }
.grid { display:grid; gap: 10px; }
.footer { position:fixed; left:0; right:0; bottom:0; border-top:1px solid
#eee; background:#fff; padding:8px; }
.map { height: 260px; border-radius: var(--radius); overflow: hidden; border:
1px solid #eee; margin-top: 10px; }
.small { font-size: 12px; color:#666 }
.bold { font-weight: 600 }
.center { text-align:center }
```

frontend/src/main.jsx

```
import React from 'react'
import { createRoot } from 'react-dom/client'
import App from './App.jsx'
import './styles.css'

createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

`frontend/src/App.jsx`

```
import React, { useMemo, useState, useEffect } from 'react'
import { GoogleMap, Marker, useJsApiLoader } from '@react-google-maps/api'
import { SERVICES, SALONS } from './data'

const currency = n => new Intl.NumberFormat('es-AR', { style: 'currency',
currency: 'ARS', maximumFractionDigits: 0 }).format(n)
const timeSlots = ['10:00','11:00','12:30','14:00','15:30','17:00','18:30']
const dayLabel = (offset=0) => { const d = new Date(); d.setDate(d.getDate()
+offset); return d.toLocaleDateString('es-AR',{weekday:'short', day:'2-
digit', month:'2-digit'}) }

export default function App(){
  const [screen, setScreen] = useState('home')
  const [query, setQuery] = useState('')
  const [filters, setFilters] = useState([])
  const [selectedSalon, setSelectedSalon] = useState(null)
  const [selectedService, setSelectedService] = useState(null)
  const [slot, setSlot] = useState('')
  const [selectedDay, setSelectedDay] = useState(0)
  const [paying, setPaying] = useState(false)

  const { isLoaded } = useJsApiLoader({ id:'google-map', googleMapsApiKey:
import.meta.env.VITE_GOOGLE_MAPS_API_KEY })

  // Si vuelve de MP con ?status=approved mostramos éxito
  useEffect(()=>{
    const sp = new URLSearchParams(location.search)
    if(sp.get('status')==='approved'){
      setScreen('success')
    }
  },[])

  const filtered = useMemo(()=>{
    const q = query.toLowerCase()
    return SALONS.filter(s => (!q || s.name.toLowerCase().includes(q) ||
s.address.toLowerCase().includes(q)) && (filters.length===0 ||
filters.every(fid=>s.services.includes(fid))))
  },[query, filters])

  const center = useMemo(()=>{
    if(!filtered.length) return { lat:-34.6037, lng:-58.3816 }
    const lat = filtered.reduce((a,s)=>a+s.lat,0)/filtered.length
    const lng = filtered.reduce((a,s)=>a+s.lng,0)/filtered.length
    return { lat, lng }
  },[filtered])
  function toggleFilter(id){ setFilters(prev => prev.includes(id) ?
prev.filter(x=>x!==id) : [...prev,id]) }

  async function pay(){
    if(!selectedSalon || !selectedService || !slot) return
    setPaying(true)
    // Guardamos la reserva local para mostrarla en Success
    localStorage.setItem('nailmeet:lastBooking', JSON.stringify({
      salon: selectedSalon, service: selectedService, dayOffset:
selectedDay, time: slot
    }))

    try{
      const res = await fetch(`${import.meta.env.VITE_API_BASE_URL}/
create_preference`,{
        method:'POST', headers:{'Content-Type':'application/json'},
        body: JSON.stringify({
          service: { id:selectedService.id, name:selectedService.name,
price:selectedService.price },
          salon: { id:selectedSalon.id, name:selectedSalon.name },
          booking: { dayOffset: selectedDay, time: slot },
          backUrl: `${import.meta.env.VITE_FRONTEND_URL}`
        })
      })
      const data = await res.json()
      if(!data.preferenceId) throw new Error('No preferenceId')
      // Abrir Checkout Pro
      const mp = new window.MercadoPago(import.meta.env.VITE_MP_PUBLIC_KEY,
{ locale: 'es-AR' })
      mp.checkout({ preference: { id: data.preferenceId }, autoOpen: true })
    }catch(e){
      alert('No se pudo iniciar el pago. Revisá el backend y las claves.')
      console.error(e)
    }finally{
      setPaying(false)
    }
  }

  return (
    <div>
      <div className="header"><div className="h1">NailMeet</div><div
className="small">Reservá en segundos</div></div>
      <div className="container">
      {screen==='home' && (
        <>
          <div className="row" style={{marginTop:8}}>
            <input className="input" placeholder="Buscar por nombre o
dirección" value={query} onChange={e=>setQuery(e.target.value)} />
            <button className="btn" onClick={()=>{ setQuery('');
setFilters([]) }}>Limpiar</button>
```

```
          </div>

          <div className="row" style={{overflowX:'auto', padding:'6px 0'}}>
            {SERVICES.map(s => (
              <button key={s.id} className={`btn $
{filters.includes(s.id)?'primary':''}`} onClick={()=>toggleFilter(s.id)}
>{s.name.split(' ')[0]}</button>
            ))}
          </div>

          <div className="map">
            {isLoaded ? (
              <GoogleMap mapContainerStyle={{height:'100%', width:'100%'}}
center={center} zoom={13}>
                {filtered.map(s=> (
                  <Marker key={s.id} position={{lat:s.lat, lng:s.lng}}
onClick={()=>{ setSelectedSalon(s); setScreen('detail') }} />
                ))}
              </GoogleMap>
            ) : (
              <div className="center" style={{padding:20}}>Cargando mapa…</
div>
            )}
          </div>

          <div className="grid" style={{marginTop:12}}>
            {filtered.map(s => (
              <div key={s.id} className="card" onClick={()=>{
setSelectedSalon(s); setScreen('detail') }} style={{cursor:'pointer'}}>
                <div className="row">
                  <img src={s.images[0]} alt={s.name} style={{width:96,
height:96, borderRadius:12, objectFit:'cover'}}/>
                  <div style={{flex:1}}>
                    <div className="bold">{s.name}</div>
                    <div className="small">{s.address}</div>
                    <div className="small">⭐ {s.rating} ({s.reviews}) •
{s.distanceKm} km</div>
                    <div className="row" style={{gap:6, marginTop:6}}>
                      {s.services.slice(0,3).map(id=> (
                        <span key={id}
className="badge">{SERVICES.find(x=>x.id===id)?.name.split(' ')[0]}</span>
                      ))}
                    </div>
                  </div>
                </div>
              </div>
            ))}
          </div>
        </>
      )}
      {screen==='detail' && selectedSalon && (
        <>
          <button className="btn" onClick={()=>setScreen('home')}>← Volver</
button>
          <h2>{selectedSalon.name}</h2>
          <div className="small">{selectedSalon.address} • ⭐
{selectedSalon.rating} ({selectedSalon.reviews})</div>
          <div className="grid" style={{gridTemplateColumns:'2fr 1fr',
marginTop:8}}>
            {selectedSalon.images.map((src,i)=> (
              <img key={i} src={src} style={{width:'100%', height:140,
objectFit:'cover', borderRadius:12}}/>
            ))}
          </div>

          <h3 style={{marginTop:12}}>Servicios</h3>
          <div className="grid">
            {selectedSalon.services.map(id=>{
              const s = SERVICES.find(x=>x.id===id)
              return (
                <div key={id} className="card row"
style={{justifyContent:'space-between', alignItems:'center'}}>
                  <div>
                    <div className="bold">{s.name}</div>
                    <div className="small">{s.duration} min •
{currency(s.price)}</div>
                  </div>
                  <button className="btn primary" onClick={()=>{
setSelectedService(s); setScreen('booking') }}>Reservar</button>
                </div>
              )
            })}
          </div>
        </>
      )}

      {screen==='booking' && selectedSalon && selectedService && (
        <>
          <button className="btn" onClick={()=>setScreen('detail')}>←
Volver</button>
          <h2>Reservar turno</h2>
          <div className="card">
            <div className="small">Salón</div>
            <div className="bold">{selectedSalon.name}</div>
            <div className="small">Servicio: {selectedService.name} •
{currency(selectedService.price)}</div>
          </div>

          <div style={{marginTop:10}}>
            <div className="bold small">Elegí un día</div>
            <div className="row" style={{overflowX:'auto', padding:'6px 0'}}>
```

```
              {Array.from({length:7}).map((_,i)=> (
                <button key={i} className={`btn $
{selectedDay===i?'primary':''}`} onClick={()=>setSelectedDay(i)}
>{dayLabel(i)}</button>
              ))}
            </div>
          </div>

          <div style={{marginTop:10}}>
            <div className="bold small">Elegí un horario</div>
            <div className="grid"
style={{gridTemplateColumns:'repeat(3,1fr)'}}>
              {timeSlots.map(t => (
                <button key={t} className={`btn ${slot===t?'primary':''}`}
onClick={()=>setSlot(t)}>{t}</button>
              ))}
            </div>
          </div>

          <div className="row" style={{marginTop:12}}>
            <button className="btn primary" disabled={!slot || paying}
onClick={pay}>{paying?'Procesando…':'Pagar y confirmar'}</button>
          </div>
          <div className="small">Si el pago se aprueba, volverás
automáticamente y verás la confirmación.</div>
        </>
      )}

      {screen==='success' && (
        <Success />
      )}
      </div>
    </div>
  )
}

function Success(){
  const booking =
JSON.parse(localStorage.getItem('nailmeet:lastBooking')||'null')
  if(!booking){
    return (<div className="container"><h2>Pago aprobado 🦜</h2><p>No se
encontró la reserva local, pero el pago fue aprobado.</p><a className="btn"
href="/">Ir al inicio</a></div>)
  }
  return (
    <div className="container" style={{textAlign:'center'}}>
      <h2>¡Turno confirmado!</h2>
      <p>{booking.service.name} en {booking.salon.name}</p>
      <p className="small">{dayLabel(booking.dayOffset)} a las
{booking.time} • Duración {booking.service.duration} min</p>
      <a className="btn" href="/">Volver al inicio</a>
    </div>
  )
}
```

`frontend/src/Success.jsx` **(opcional)**

> Ya está inline en `App.jsx`. Podés extraerlo si preferís componentes separados.

🧰Backend (Node.js + Express + Mercado Pago)

`backend/package.json`

```
{
  "name": "nailmeet-backend",
  "private": true,
  "version": "1.0.0",
  "type": "module",
  "scripts": {
    "dev": "node server.js",
    "start": "node server.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "mercadopago": "^2.1.20"
  }
}
```

`backend/.env.example`

```
PORT=3001
MP_ACCESS_TOKEN=TEST-xxxxxxxxxxxxxxxxxxxxxxxxx
FRONTEND_URL=http://localhost:5173
```

> Copiá a `.env` y completá.

`backend/server.js`

```
import express from 'express'
import cors from 'cors'
import dotenv from 'dotenv'
import mercadopago from 'mercadopago'

dotenv.config()
const app = express()
```

```
app.use(express.json())
app.use(cors({ origin: [process.env.FRONTEND_URL], credentials: true }))

// Configurar credenciales de Mercado Pago
mercadopago.configure({ access_token: process.env.MP_ACCESS_TOKEN })

// Crear preferencia
app.post('/create_preference', async (req,res) => {
  try{
    const { service, salon, booking, backUrl } = req.body
    const success = `${backUrl || process.env.FRONTEND_URL}?status=approved`
    const failure = `${backUrl || process.env.FRONTEND_URL}?status=failure`
    const pending = `${backUrl || process.env.FRONTEND_URL}?status=pending`

    const preference = {
      items: [
        { title: `${service.name} - ${salon.name}`, unit_price:
Number(service.price), quantity: 1 }
      ],
      metadata: { salonId: salon.id, serviceId: service.id, dayOffset:
booking.dayOffset, time: booking.time },
      back_urls: { success, failure, pending },
      auto_return: 'approved'
    }

    const response = await mercadopago.preferences.create(preference)
    return res.json({ preferenceId: response.body.id })
  }catch(e){
    console.error(e)
    return res.status(500).json({ error: 'Error creando preferencia' })
  }
})

// Webhook de notificaciones (opcional, para confirmar del lado servidor)
app.post('/webhook', async (req,res) => {
  // Mercado Pago enviará notificaciones aquí. Para pruebas basta con
responder 200.
  console.log('Webhook:', JSON.stringify(req.body))
  res.sendStatus(200)
})

const port = process.env.PORT || 3001
app.listen(port, () => console.log(`API escuchando en http://localhost:$
{port}`))
```

🧪 Cómo correr localmente

1) **Backend**

```
cd backend
cp .env.example .env   # completá MP_ACCESS_TOKEN y FRONTEND_URL
npm i
npm run dev
```

La API queda en `http://localhost:3001`

2) **Frontend**

```
cd ../frontend
cp .env.example .env   # completá las 3 variables
npm i
npm run dev
```

La web queda en `http://localhost:5173`

🧦 Deploy sencillo

Backend en Render (gratis)

1. Subí la carpeta `backend` a un repo de GitHub.
2. En Render → **New Web Service** → conectá el repo.
3. **Build Command**: `npm i`
4. **Start Command**: `npm start`
5. **Environment**: agregá `MP_ACCESS_TOKEN` y `FRONTEND_URL` (tu dominio de Vercel).

Frontend en Vercel (gratis)

1. Subí la carpeta `frontend` a otro repo (o subcarpeta del mismo).
2. En Vercel → **New Project** → importá repo.
3. **Framework**: Vite.
4. Variables de entorno: `VITE_GOOGLE_MAPS_API_KEY`, `VITE_MP_PUBLIC_KEY`, `VITE_API_BASE_URL` (URL del backend en Render), `VITE_FRONTEND_URL` (tu dominio de Vercel).
5. Deploy 💫

🔗 Checklist para que todo funcione

- [ ] `MP_ACCESS_TOKEN` (sandbox) válido en el backend.
- [ ] `VITE_MP_PUBLIC_KEY` (sandbox) válido en el frontend.
- [ ] API del backend **accesible públicamente** (Render/otro) y usada en `VITE_API_BASE_URL`.
- [ ] **Google Maps API Key** habilitada para Maps JavaScript API.
- [ ] `FRONTEND_URL` y `VITE_FRONTEND_URL` apuntando a tu dominio final.

- [ ] Probar pago: debería redirigir a `?status=approved` y mostrar la confirmación con los datos guardados.

🛠️Notas y mejoras sugeridas

- Implementar **almacenamiento real** de la reserva en el backend (DB o Firestore) y confirmar vía **webhook**.
- Validar disponibilidad real de horarios (ahora es mock).
- Agregar autenticación ligera (p. ej., Magic Link o Firebase Auth) para historial de turnos.
- Reemplazar imágenes y textos por datos reales de tus salones.

Cualquier duda con el deploy o las claves, decime y lo ajustamos al toque.