Guia de Implementação - Aplicativo de Agendamento Dani Tattoos

Este guia fornece instruções detalhadas para implementar as funcionalidades restantes do aplicativo de agendamento para Dani Tattoos.

1. Integração com Google Calendar

1.1 Configuração do OAuth

1. Acesse o Google Cloud Console
2. Crie um novo projeto
3. Ative a API do Google Calendar
4. Configure as credenciais OAuth:
5. Tipo: Aplicativo Web
6. Origens JavaScript autorizadas: URL do seu aplicativo (ex: https://app-agenda.vercel.app)
7. URIs de redirecionamento: URL do seu aplicativo + /api/auth/callback/google

```
 // src/lib/googleAuth.js
import { google } from 'googleapis';

// Configure o cliente OAuth2
const oauth2Client = new google.auth.OAuth2(
  process.env.GOOGLE_CLIENT_ID,
  process.env.GOOGLE_CLIENT_SECRET,
  process.env.GOOGLE_REDIRECT_URI
);

// Função para gerar URL de autorização
export function getAuthUrl() {
  return oauth2Client.generateAuthUrl({
    access_type: 'offline',
    scope: ['https://www.googleapis.com/auth/calendar']
  });
}

// Função para obter tokens de acesso
export async function getTokens(code) {
  const { tokens } = await oauth2Client.getToken(code);
  return tokens;
}
```

```
// Função para configurar cliente com tokens
export function setCredentials(tokens) {
  oauth2Client.setCredentials(tokens);
  return oauth2Client;
}
```

1.2 Implementação da API do Calendar

```
 // src/lib/calendarApi.js
import { google } from 'googleapis';

// Função para listar eventos
export async function listEvents(auth, calendarId, timeMin, timeMax) {
  const calendar = google.calendar({ version: 'v3', auth });
  const response = await calendar.events.list({
    calendarId,
    timeMin,
    timeMax,
    singleEvents: true,
    orderBy: 'startTime',
  });
  return response.data.items;
}

// Função para criar evento
export async function createEvent(auth, calendarId, event) {
  const calendar = google.calendar({ version: 'v3', auth });
  const response = await calendar.events.insert({
    calendarId,
    resource: event,
  });
  return response.data;
}

// Função para atualizar evento
export async function updateEvent(auth, calendarId, eventId, event) {
  const calendar = google.calendar({ version: 'v3', auth });
  const response = await calendar.events.update({
    calendarId,
    eventId,
    resource: event,
  });
  return response.data;
}

// Função para excluir evento
export async function deleteEvent(auth, calendarId, eventId) {
  const calendar = google.calendar({ version: 'v3', auth });
  await calendar.events.delete({
```

```
    calendarId,
    eventId,
  });
  return true;
}
```

1.3 Componente de Seleção de Agenda (SP ou PP)

```
// src/components/CalendarSelector.js
import React, { useState, useEffect } from 'react';

export default function CalendarSelector({ onSelect }) {
  const [calendars, setCalendars] = useState([]);
  const [selectedCalendar, setSelectedCalendar] = useState('');
  const [calendarType, setCalendarType] = useState('SP'); // SP ou PP

  useEffect(() => {
    // Carregar calendários do usuário
    async function loadCalendars() {
      try {
        const response = await fetch('/api/calendars');
        const data = await response.json();
        setCalendars(data.calendars);
      } catch (error) {
        console.error('Erro ao carregar calendários:', error);
      }
    }

    loadCalendars();
  }, []);

  const handleCalendarChange = (e) => {
    setSelectedCalendar(e.target.value);
    onSelect(e.target.value, calendarType);
  };

  const handleTypeChange = (e) => {
    setCalendarType(e.target.value);
    onSelect(selectedCalendar, e.target.value);
  };

  return (
    <div className="calendar-selector">
      <div className="form-group">
        <label>Tipo de Agenda:</label>
        <select value={calendarType} onChange={handleTypeChange}>
          <option value="SP">SP</option>
          <option value="PP">PP</option>
        </select>
      </div>
```

```
      <div className="form-group">
        <label>Selecione o Calendário:</label>
        <select value={selectedCalendar} onChange={handleCalendarChange}>
          <option value="">Selecione...</option>
          {calendars.map(calendar => (
            <option key={calendar.id} value={calendar.id}>
              {calendar.summary}
            </option>
          ))}
        </select>
      </div>
    </div>
  );
}
```

2. Interface de Agendamento

2.1 Visualização Mensal do Calendário

```
// src/components/MonthlyCalendar.js
import React, { useState, useEffect } from 'react';
import FullCalendar from '@fullcalendar/react';
import dayGridPlugin from '@fullcalendar/daygrid';
import interactionPlugin from '@fullcalendar/interaction';

export default function MonthlyCalendar({ onDateClick, events }) {
  const handleDateClick = (info) => {
    onDateClick(info.date);
  };

  return (
    <div className="monthly-calendar">
      <FullCalendar
        plugins={[dayGridPlugin, interactionPlugin]}
        initialView="dayGridMonth"
        locale="pt-br"
        headerToolbar={{
          left: 'prev,next today',
          center: 'title',
          right: 'dayGridMonth'
        }}
        events={events}
        dateClick={handleDateClick}
        height="auto"
      />
    </div>
```

```
  );
}
```

2.2 Formulário de Agendamento

```
// src/components/AppointmentForm.js
import React, { useState } from 'react';

export default function AppointmentForm({ selectedDate, onSubmit,
calendarType }) {
  const [formData, setFormData] = useState({
    clientName: '',
    date: selectedDate ? selectedDate.toISOString().split('T')[0] : '',
    time: '10:00',
    projectValue: '',
    deposit: '',
    description: ''
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({
      ...prev,
      [name]: value
    }));
  };

  const calculateRemaining = () => {
    const projectValue = parseFloat(formData.projectValue) || 0;
    const deposit = parseFloat(formData.deposit) || 0;
    return (projectValue - deposit).toFixed(2);
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    // Criar objeto de evento para o Google Calendar
    const event = {
      summary: formData.clientName, // Nome do cliente como título
      description: formData.description,
      start: {
        dateTime: `${formData.date}T${formData.time}:00`,
        timeZone: 'America/Sao_Paulo',
      },
      end: {
        dateTime: `${formData.date}T${formData.time}:00`,
        timeZone: 'America/Sao_Paulo',
      },
      extendedProperties: {
        private: {
```

```
        projectValue: formData.projectValue,
        deposit: formData.deposit,
        remaining: calculateRemaining(),
        calendarType: calendarType // SP ou PP
      }
    }
  };

  onSubmit(event);
};

return (
  <form onSubmit={handleSubmit} className="appointment-form">
    <h2>Novo Agendamento</h2>

    <div className="form-group">
      <label htmlFor="clientName">Nome do Cliente:</label>
      <input
        type="text"
        id="clientName"
        name="clientName"
        value={formData.clientName}
        onChange={handleChange}
        required
      />
    </div>

    <div className="form-group">
      <label htmlFor="date">Data:</label>
      <input
        type="date"
        id="date"
        name="date"
        value={formData.date}
        onChange={handleChange}
        required
      />
    </div>

    <div className="form-group">
      <label htmlFor="time">Horário:</label>
      <input
        type="time"
        id="time"
        name="time"
        value={formData.time}
        onChange={handleChange}
        required
      />
    </div>

    <div className="form-group">
```

```
        <label htmlFor="projectValue">Valor do Projeto (R$):</label>
        <input
          type="number"
          id="projectValue"
          name="projectValue"
          value={formData.projectValue}
          onChange={handleChange}
          step="0.01"
          min="0"
          required
        />
      </div>

      <div className="form-group">
        <label htmlFor="deposit">Sinal (R$):</label>
        <input
          type="number"
          id="deposit"
          name="deposit"
          value={formData.deposit}
          onChange={handleChange}
          step="0.01"
          min="0"
          required
        />
      </div>

      <div className="form-group">
        <label>Valor Restante (R$):</label>
        <div className="calculated-value">{calculateRemaining()}</div>
      </div>

      <div className="form-group">
        <label htmlFor="description">Descrição do Projeto:</label>
        <textarea
          id="description"
          name="description"
          value={formData.description}
          onChange={handleChange}
          rows="4"
        ></textarea>
      </div>

      <button type="submit" className="submit-button">Salvar Agendamento</
button>
    </form>
  );
}
```

3. Geração de Comprovantes

3.1 Componente de Geração de Comprovante

```
// src/components/ReceiptGenerator.js
import React, { useRef } from 'react';
import html2canvas from 'html2canvas';

export default function ReceiptGenerator({ appointment, calendarType }) {
  const receiptRef = useRef(null);

  const generateReceipt = async () => {
    if (!receiptRef.current) return;

    try {
      const canvas = await html2canvas(receiptRef.current);
      const image = canvas.toDataURL('image/png');

      // Criar link para download
      const link = document.createElement('a');
      link.href = image;
      link.download = `comprovante-${appointment.clientName.replace(/\s+/g,
'-')}.png`;

      // Compartilhar via menu de contexto do dispositivo
      if (navigator.share) {
        const blob = await (await fetch(image)).blob();
        const file = new File([blob], link.download, { type: 'image/png' });

        navigator.share({
          title: 'Comprovante de Agendamento',
          text: `Comprovante de agendamento para ${appointment.clientName}`,
          files: [file]
        }).catch(error => {
          console.log('Erro ao compartilhar:', error);
          // Fallback para download direto
          link.click();
        });
      } else {
        // Fallback para download direto
        link.click();
      }
    } catch (error) {
      console.error('Erro ao gerar comprovante:', error);
    }
  };

  // Template para SP
  const SPTemplate = () => (
    <div className="receipt sp-receipt" ref={receiptRef}>
```

```
      <div className="receipt-header">
        <img src="/logo.png" alt="Dani Tattoos Logo" className="receipt-logo" />
        <h2>Dani Tattoos - SP</h2>
      </div>

      <div className="receipt-content">
        <div className="receipt-field">
          <span className="label">Cliente:</span>
          <span className="value">{appointment.clientName}</span>
        </div>

        <div className="receipt-field">
          <span className="label">Data:</span>
          <span className="value">{new
Date(appointment.date).toLocaleDateString('pt-BR')}</span>
        </div>

        <div className="receipt-field">
          <span className="label">Horário:</span>
          <span className="value">{appointment.time}</span>
        </div>

        <div className="receipt-field">
          <span className="label">Valor do Projeto:</span>
          <span className="value">R$
{parseFloat(appointment.projectValue).toFixed(2)}</span>
        </div>

        <div className="receipt-field">
          <span className="label">Sinal:</span>
          <span className="value">R$ {parseFloat(appointment.deposit).toFixed(2)}</
span>
        </div>

        <div className="receipt-field">
          <span className="label">Valor Restante:</span>
          <span className="value">R$ {(parseFloat(appointment.projectValue) -
parseFloat(appointment.deposit)).toFixed(2)}</span>
        </div>

        <div className="receipt-description">
          <span className="label">Descrição:</span>
          <p>{appointment.description}</p>
        </div>
      </div>

      <div className="receipt-footer">
        <p>Obrigado pela preferência!</p>
        <p>Dani Tattoos - São Paulo</p>
      </div>
    </div>
  );
```

```
  // Template para PP
  const PPTemplate = () => (
    <div className="receipt pp-receipt" ref={receiptRef}>
      <div className="receipt-header">
        <img src="/logo.png" alt="Dani Tattoos Logo" className="receipt-logo" />
        <h2>Dani Tattoos - PP</h2>
      </div>

      <div className="receipt-content">
        <div className="receipt-field">
          <span className="label">Cliente:</span>
          <span className="value">{appointment.clientName}</span>
        </div>

        <div className="receipt-field">
          <span className="label">Data:</span>
          <span className="value">{new
Date(appointment.date).toLocaleDateString('pt-BR')}</span>
        </div>

        <div className="receipt-field">
          <span className="label">Horário:</span>
          <span className="value">{appointment.time}</span>
        </div>

        <div className="receipt-field">
          <span className="label">Valor do Projeto:</span>
          <span className="value">R$
{parseFloat(appointment.projectValue).toFixed(2)}</span>
        </div>

        <div className="receipt-field">
          <span className="label">Sinal:</span>
          <span className="value">R$ {parseFloat(appointment.deposit).toFixed(2)}</
span>
        </div>

        <div className="receipt-field">
          <span className="label">Valor Restante:</span>
          <span className="value">R$ {(parseFloat(appointment.projectValue) -
parseFloat(appointment.deposit)).toFixed(2)}</span>
        </div>

        <div className="receipt-description">
          <span className="label">Descrição:</span>
          <p>{appointment.description}</p>
        </div>
      </div>

      <div className="receipt-footer">
        <p>Obrigado pela preferência!</p>
```

```
        <p>Dani Tattoos - Praia Preta</p>
      </div>
    </div>
  );

  return (
    <div className="receipt-generator">
      {calendarType === 'SP' ? <SPTemplate /> : <PPTemplate />}

      <button onClick={generateReceipt} className="share-button">
        Gerar e Compartilhar Comprovante
      </button>
    </div>
  );
}
```

3.2 Estilização dos Comprovantes

```
/* src/styles/receipt.css */
.receipt {
  width: 100%;
  max-width: 500px;
  padding: 20px;
  border: 1px solid #ddd;
  border-radius: 8px;
  background-color: white;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  margin: 0 auto;
}

.receipt-header {
  display: flex;
  align-items: center;
  margin-bottom: 20px;
  border-bottom: 2px solid #e74c3c;
  padding-bottom: 10px;
}

.receipt-logo {
  width: 60px;
  height: 60px;
  margin-right: 15px;
}

.receipt-header h2 {
  color: #e74c3c;
  margin: 0;
  font-size: 1.5rem;
}
```

```css
.receipt-content {
  margin-bottom: 20px;
}

.receipt-field {
  display: flex;
  justify-content: space-between;
  margin-bottom: 10px;
  padding-bottom: 5px;
  border-bottom: 1px dashed #eee;
}

.label {
  font-weight: bold;
  color: #333;
}

.value {
  color: #555;
}

.receipt-description {
  margin-top: 15px;
}

.receipt-description p {
  margin-top: 5px;
  padding: 10px;
  background-color: #f9f9f9;
  border-radius: 4px;
}

.receipt-footer {
  margin-top: 20px;
  text-align: center;
  font-size: 0.9rem;
  color: #777;
  border-top: 1px solid #eee;
  padding-top: 15px;
}

/* Estilos específicos para SP */
.sp-receipt {
  background-color: #fff9f9;
}

.sp-receipt .receipt-header {
  border-bottom-color: #e74c3c;
}

.sp-receipt .receipt-header h2 {
  color: #e74c3c;
```

```
}

/* Estilos específicos para PP */
.pp-receipt {
  background-color: #f9f9ff;
}

.pp-receipt .receipt-header {
  border-bottom-color: #3498db;
}

.pp-receipt .receipt-header h2 {
  color: #3498db;
}

.share-button {
  display: block;
  width: 100%;
  max-width: 500px;
  margin: 20px auto;
  padding: 12px;
  background-color: #e74c3c;
  color: white;
  border: none;
  border-radius: 4px;
  font-size: 1rem;
  cursor: pointer;
  transition: background-color 0.3s;
}

.share-button:hover {
  background-color: #c0392b;
}
```

4. Compartilhamento via WhatsApp

4.1 Implementação do Botão de Compartilhamento

```
// src/components/ShareButton.js
import React from 'react';

export default function ShareButton({ imageUrl, clientName }) {
  const handleShare = async () => {
    if (navigator.share) {
      try {
        // Obter o blob da imagem
        const response = await fetch(imageUrl);
        const blob = await response.blob();
```

```
        const file = new File([blob], `comprovante-${clientName.replace(/\s+/g,
'-')}.png`, { type: 'image/png' });

        // Compartilhar via API Web Share
        await navigator.share({
          title: 'Comprovante de Agendamento',
          text: `Comprovante de agendamento para ${clientName}`,
          files: [file]
        });
      } catch (error) {
        console.error('Erro ao compartilhar:', error);

        // Fallback para WhatsApp direto se a API Web Share falhar
        const whatsappUrl = `https://wa.me/?text=$
{encodeURIComponent(`Comprovante de agendamento para ${clientName}`)}`;
        window.open(whatsappUrl, '_blank');
      }
    } else {
      // Fallback para navegadores que não suportam a API Web Share
      const whatsappUrl = `https://wa.me/?text=$
{encodeURIComponent(`Comprovante de agendamento para ${clientName}`)}`;
      window.open(whatsappUrl, '_blank');
    }
  };

  return (
    <button onClick={handleShare} className="share-button">
      Compartilhar via WhatsApp
    </button>
  );
}
```

5. Configuração do Ambiente

5.1 Variáveis de Ambiente

Crie um arquivo `.env.local` na raiz do projeto com as seguintes variáveis:

```
 GOOGLE_CLIENT_ID=seu-client-id
GOOGLE_CLIENT_SECRET=seu-client-secret
GOOGLE_REDIRECT_URI=https://seu-app.vercel.app/api/auth/callback/google
NEXTAUTH_URL=https://seu-app.vercel.app
NEXTAUTH_SECRET=uma-string-secreta-aleatoria
```

5.2 API Routes para Autenticação

```
// src/pages/api/auth/[...nextauth].js
import NextAuth from 'next-auth';
import GoogleProvider from 'next-auth/providers/google';

export default NextAuth({
  providers: [
    GoogleProvider({
      clientId: process.env.GOOGLE_CLIENT_ID,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET,
      authorization: {
        params: {
          scope: 'https://www.googleapis.com/auth/userinfo.profile https://
www.googleapis.com/auth/userinfo.email https://www.googleapis.com/auth/
calendar',
          prompt: 'consent',
          access_type: 'offline',
          response_type: 'code'
        }
      }
    })
  ],
  callbacks: {
    async jwt({ token, account }) {
      // Persistir os tokens de acesso e atualização no token JWT
      if (account) {
        token.accessToken = account.access_token;
        token.refreshToken = account.refresh_token;
        token.expiresAt = account.expires_at;
      }
      return token;
    },
    async session({ session, token }) {
      // Passar os tokens para o cliente
      session.accessToken = token.accessToken;
      return session;
    }
  },
  pages: {
    signIn: '/auth/signin',
  }
});
```

5.3 API Routes para Calendários

```
// src/pages/api/calendars.js
import { getServerSession } from 'next-auth/next';
import { google } from 'googleapis';
```

```
import { authOptions } from './auth/[...nextauth]';

export default async function handler(req, res) {
  const session = await getServerSession(req, res, authOptions);

  if (!session) {
    return res.status(401).json({ error: 'Não autenticado' });
  }

  try {
    const oauth2Client = new google.auth.OAuth2(
      process.env.GOOGLE_CLIENT_ID,
      process.env.GOOGLE_CLIENT_SECRET,
      process.env.GOOGLE_REDIRECT_URI
    );

    oauth2Client.setCredentials({
      access_token: session.accessToken
    });

    const calendar = google.calendar({ version: 'v3', auth: oauth2Client });
    const response = await calendar.calendarList.list();

    res.status(200).json({ calendars: response.data.items });
  } catch (error) {
    console.error('Erro ao obter calendários:', error);
    res.status(500).json({ error: 'Erro ao obter calendários' });
  }
}
```

6. Integração de Componentes

6.1 Página Principal de Agendamento

```
// src/pages/agenda.js
import React, { useState, useEffect } from 'react';
import { useSession, signIn } from 'next-auth/react';
import MonthlyCalendar from '../components/MonthlyCalendar';
import AppointmentForm from '../components/AppointmentForm';
import CalendarSelector from '../components/CalendarSelector';
import ReceiptGenerator from '../components/ReceiptGenerator';

export default function AgendaPage() {
  const { data: session, status } = useSession();
  const [selectedDate, setSelectedDate] = useState(null);
  const [events, setEvents] = useState([]);
  const [selectedCalendar, setSelectedCalendar] = useState('');
  const [calendarType, setCalendarType] = useState('SP');
```

```
const [showForm, setShowForm] = useState(false);
const [showReceipt, setShowReceipt] = useState(false);
const [currentAppointment, setCurrentAppointment] = useState(null);

// Verificar autenticação
useEffect(() => {
  if (status === 'unauthenticated') {
    signIn('google');
  }
}, [status]);

// Carregar eventos quando o calendário for selecionado
useEffect(() => {
  if (session && selectedCalendar) {
    fetchEvents();
  }
}, [session, selectedCalendar]);

// Buscar eventos do Google Calendar
const fetchEvents = async () => {
  try {
    const response = await fetch(`/api/events?calendarId=${selectedCalendar}`);
    const data = await response.json();

    // Formatar eventos para o FullCalendar
    const formattedEvents = data.events.map(event => ({
      id: event.id,
      title: event.summary,
      start: event.start.dateTime || event.start.date,
      end: event.end.dateTime || event.end.date,
      extendedProps: event.extendedProperties?.private || {}
    }));

    setEvents(formattedEvents);
  } catch (error) {
    console.error('Erro ao buscar eventos:', error);
  }
};

// Manipular seleção de data no calendário
const handleDateClick = (date) => {
  setSelectedDate(date);
  setShowForm(true);
  setShowReceipt(false);
};

// Manipular seleção de calendário
const handleCalendarSelect = (calendarId, type) => {
  setSelectedCalendar(calendarId);
  setCalendarType(type);
};
```

```
  // Manipular envio do formulário
  const handleFormSubmit = async (eventData) => {
    try {
      const response = await fetch('/api/events', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          calendarId: selectedCalendar,
          event: eventData
        }),
      });

      const data = await response.json();

      if (response.ok) {
        // Extrair dados do evento para o comprovante
        const appointment = {
          clientName: eventData.summary,
          date: eventData.start.dateTime,
          time: new Date(eventData.start.dateTime).toLocaleTimeString('pt-BR', { hour:
'2-digit', minute: '2-digit' }),
          projectValue: eventData.extendedProperties.private.projectValue,
          deposit: eventData.extendedProperties.private.deposit,
          description: eventData.description
        };

        setCurrentAppointment(appointment);
        setShowForm(false);
        setShowReceipt(true);

        // Atualizar lista de eventos
        fetchEvents();
      }
    } catch (error) {
      console.error('Erro ao criar evento:', error);
    }
  };

  if (status === 'loading') {
    return <div className="loading">Carregando...</div>;
  }

  return (
    <div className="agenda-page">
      <h1>Gerenciamento de Agendamentos</h1>

      <CalendarSelector onSelect={handleCalendarSelect} />

      {selectedCalendar && (
        <MonthlyCalendar
```

```
          onDateClick={handleDateClick}
          events={events}
        />
      )}

      {showForm && (
        <AppointmentForm
          selectedDate={selectedDate}
          onSubmit={handleFormSubmit}
          calendarType={calendarType}
        />
      )}

      {showReceipt && currentAppointment && (
        <ReceiptGenerator
          appointment={currentAppointment}
          calendarType={calendarType}
        />
      )}
    </div>
  );
}
```

7. Estilização e Responsividade

7.1 Estilos Globais

```
/* src/styles/globals.css */
:root {
  --primary-color: #e74c3c;
  --secondary-color: #2c3e50;
  --text-color: #333;
  --light-color: #f8f9fa;
  --dark-color: #212529;
}

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

body {
  background-color: var(--light-color);
  color: var(--text-color);
  line-height: 1.6;
}
```

```css
.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 1rem;
}

h1, h2, h3 {
  color: var(--primary-color);
  margin-bottom: 1rem;
}

.form-group {
  margin-bottom: 1rem;
}

label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 500;
}

input, select, textarea {
  width: 100%;
  padding: 0.75rem;
  border: 1px solid #ddd;
  border-radius: 4px;
  font-size: 1rem;
}

button {
  background-color: var(--primary-color);
  color: white;
  border: none;
  padding: 0.75rem 1.5rem;
  border-radius: 4px;
  cursor: pointer;
  font-size: 1rem;
  transition: background-color 0.3s;
}

button:hover {
  background-color: #c0392b;
}

.loading {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  font-size: 1.2rem;
  color: var(--primary-color);
```

```
}

/* Responsividade */
@media (max-width: 768px) {
  .container {
    padding: 0.5rem;
  }

  h1 {
    font-size: 1.5rem;
  }

  h2 {
    font-size: 1.3rem;
  }

  input, select, textarea {
    padding: 0.5rem;
  }

  button {
    padding: 0.5rem 1rem;
  }
}
```

8. Configuração do PWA

8.1 Atualização do Manifesto

```
// public/manifest.json
{
  "name": "Dani Tattoos - Agendamento",
  "short_name": "Dani Tattoos",
  "description": "Aplicativo de agendamento para tatuadores",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#e74c3c",
  "icons": [
    {
      "src": "/icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
```

```
    },
    {
      "src": "/icons/icon-180x180.png",
      "sizes": "180x180",
      "type": "image/png",
      "purpose": "apple-touch-icon"
    }
  ]
}
```

8.2 Configuração do Service Worker

```
// next.config.js
const withPWA = require('next-pwa')({
  dest: 'public',
  register: true,
  skipWaiting: true,
  disable: process.env.NODE_ENV === 'development'
});

module.exports = withPWA({
  reactStrictMode: true,
  images: {
    domains: ['localhost', 'vercel.app'],
  },
});
```

9. Implantação

9.1 Configuração da Vercel

1. Faça login na Vercel
2. Importe o repositório do GitHub
3. Configure as variáveis de ambiente:
4. GOOGLE_CLIENT_ID
5. GOOGLE_CLIENT_SECRET
6. GOOGLE_REDIRECT_URI
7. NEXTAUTH_URL
8. NEXTAUTH_SECRET
9. Deploy

9.2 Configuração do Google Cloud Console

1. Acesse https://console.cloud.google.com/

2. Crie um novo projeto
3. Ative a API do Google Calendar
4. Configure as credenciais OAuth:
5. Tipo: Aplicativo Web
6. Origens JavaScript autorizadas: URL do seu aplicativo (ex: https://app-agenda.vercel.app)
7. URIs de redirecionamento: URL do seu aplicativo + /api/auth/callback/google
8. Anote o Client ID e Client Secret para usar nas variáveis de ambiente

10. Considerações Finais

Este guia fornece as instruções para implementar todas as funcionalidades solicitadas para o aplicativo de agendamento Dani Tattoos. A implementação completa requer:

1. Configuração do ambiente de desenvolvimento
2. Implementação dos componentes React
3. Configuração da autenticação OAuth com Google
4. Implementação das APIs para integração com Google Calendar
5. Desenvolvimento dos templates de comprovantes
6. Configuração do compartilhamento via WhatsApp
7. Implantação na Vercel

Siga as instruções passo a passo para completar o desenvolvimento do aplicativo. Se encontrar dificuldades, consulte a documentação oficial das bibliotecas utilizadas ou entre em contato para obter suporte adicional.