**Sistema Body Mapping Interattivo per Documentazione Zone Trattamento**

**1. ARCHITETTURA DEL SISTEMA BODY MAPPING**

**1.1 Componenti Principali**

class BodyMappingSystem {

constructor() {

this.views = ['anterior', 'posterior', 'lateral\_dx', 'lateral\_sx'];

this.bodyParts = this.loadBodyPartsDatabase();

this.markers = [];

this.annotations = [];

this.layers = {

skeletal: false,

muscular: false,

nervous: false,

vascular: false

};

}

// Database anatomico con coordinate precise

bodyRegions = {

head: {

skull: {

areas: ['frontal', 'parietal', 'temporal', 'occipital'],

coordinates: { x: [45, 55], y: [5, 15] }

},

face: {

areas: ['tmj', 'maxilla', 'mandible', 'orbital'],

coordinates: { x: [45, 55], y: [10, 20] }

}

},

spine: {

cervical: {

vertebrae: ['C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7'],

coordinates: { x: [48, 52], y: [20, 28] }

},

thoracic: {

vertebrae: ['T1', 'T2', 'T3', 'T4', 'T5', 'T6', 'T7', 'T8', 'T9', 'T10', 'T11', 'T12'],

coordinates: { x: [48, 52], y: [28, 45] }

},

lumbar: {

vertebrae: ['L1', 'L2', 'L3', 'L4', 'L5'],

coordinates: { x: [48, 52], y: [45, 55] }

},

sacral: {

areas: ['sacrum', 'coccyx'],

coordinates: { x: [48, 52], y: [55, 60] }

}

},

upperExtremity: {

shoulder: {

right: {

areas: ['deltoid', 'rotator\_cuff', 'ac\_joint', 'glenohumeral'],

coordinates: { x: [30, 40], y: [25, 35] }

},

left: {

areas: ['deltoid', 'rotator\_cuff', 'ac\_joint', 'glenohumeral'],

coordinates: { x: [60, 70], y: [25, 35] }

}

},

arm: {

right: {

areas: ['biceps', 'triceps', 'brachialis'],

coordinates: { x: [28, 38], y: [35, 45] }

},

left: {

areas: ['biceps', 'triceps', 'brachialis'],

coordinates: { x: [62, 72], y: [35, 45] }

}

},

elbow: {

right: { coordinates: { x: [30, 36], y: [43, 47] } },

left: { coordinates: { x: [64, 70], y: [43, 47] } }

},

forearm: {

right: {

areas: ['flexors', 'extensors'],

coordinates: { x: [28, 36], y: [47, 55] }

},

left: {

areas: ['flexors', 'extensors'],

coordinates: { x: [64, 72], y: [47, 55] }

}

},

hand: {

right: {

areas: ['carpal', 'metacarpal', 'phalanges'],

coordinates: { x: [25, 35], y: [55, 62] }

},

left: {

areas: ['carpal', 'metacarpal', 'phalanges'],

coordinates: { x: [65, 75], y: [55, 62] }

}

}

},

lowerExtremity: {

hip: {

right: { coordinates: { x: [40, 48], y: [55, 62] } },

left: { coordinates: { x: [52, 60], y: [55, 62] } }

},

thigh: {

right: {

areas: ['quadriceps', 'hamstrings', 'adductors', 'it\_band'],

coordinates: { x: [38, 46], y: [62, 75] }

},

left: {

areas: ['quadriceps', 'hamstrings', 'adductors', 'it\_band'],

coordinates: { x: [54, 62], y: [62, 75] }

}

},

knee: {

right: { coordinates: { x: [40, 45], y: [73, 77] } },

left: { coordinates: { x: [55, 60], y: [73, 77] } }

},

leg: {

right: {

areas: ['tibialis', 'gastrocnemius', 'soleus'],

coordinates: { x: [39, 45], y: [77, 88] }

},

left: {

areas: ['tibialis', 'gastrocnemius', 'soleus'],

coordinates: { x: [55, 61], y: [77, 88] }

}

},

foot: {

right: {

areas: ['tarsal', 'metatarsal', 'phalanges', 'plantar\_fascia'],

coordinates: { x: [38, 46], y: [88, 95] }

},

left: {

areas: ['tarsal', 'metatarsal', 'phalanges', 'plantar\_fascia'],

coordinates: { x: [54, 62], y: [88, 95] }

}

}

}

};

// Tipi di marcatori disponibili

markerTypes = {

pain: {

icon: '✕',

color: '#FF0000',

name: 'Dolore',

description: 'Punto di dolore principale'

},

trigger\_point: {

icon: '●',

color: '#FF6B6B',

name: 'Trigger Point',

description: 'Punto trigger miofasciale'

},

treatment\_area: {

icon: '○',

color: '#4ECDC4',

name: 'Area Trattamento',

description: 'Zona da trattare'

},

inflammation: {

icon: '▲',

color: '#FFD93D',

name: 'Infiammazione',

description: 'Area infiammata'

},

surgery\_scar: {

icon: '—',

color: '#6C5CE7',

name: 'Cicatrice',

description: 'Cicatrice chirurgica'

},

radiating\_pain: {

icon: '↗',

color: '#FFA500',

name: 'Dolore Irradiato',

description: 'Direzione irradiazione'

},

numbness: {

icon: '░',

color: '#95A5A6',

name: 'Parestesia',

description: 'Area con alterazione sensitiva'

},

swelling: {

icon: '◎',

color: '#3498DB',

name: 'Edema',

description: 'Gonfiore/Edema'

}

};

addMarker(x, y, type, view, notes = '') {

const marker = {

id: generateId(),

x: x,

y: y,

type: type,

view: view,

timestamp: new Date(),

therapist: getCurrentUser(),

notes: notes,

anatomicalLocation: this.detectAnatomicalLocation(x, y, view)

};

this.markers.push(marker);

this.saveToPatientRecord(marker);

return marker;

}

detectAnatomicalLocation(x, y, view) {

// Algoritmo per determinare la zona anatomica dalle coordinate

for (const [region, parts] of Object.entries(this.bodyRegions)) {

for (const [part, data] of Object.entries(parts)) {

if (data.coordinates) {

if (x >= data.coordinates.x[0] && x <= data.coordinates.x[1] &&

y >= data.coordinates.y[0] && y <= data.coordinates.y[1]) {

return { region, part, specific: this.getSpecificArea(x, y, data) };

}

}

}

}

return { region: 'other', part: 'unspecified' };

}

}

**1.2 Sistema di Disegno e Annotazione**

class BodyDrawingSystem:

def \_\_init\_\_(self):

self.drawing\_tools = {

'point': PointTool(),

'circle': CircleTool(),

'rectangle': RectangleTool(),

'polygon': PolygonTool(),

'freehand': FreehandTool(),

'arrow': ArrowTool(),

'text': TextTool()

}

self.pain\_patterns = self.load\_pain\_patterns()

def create\_pain\_map(self, patient\_id, session\_id):

"""

Crea mappa del dolore interattiva

"""

pain\_map = {

'patient\_id': patient\_id,

'session\_id': session\_id,

'date': datetime.now(),

'therapist': self.get\_current\_therapist(),

'views': {

'anterior': [],

'posterior': [],

'lateral\_right': [],

'lateral\_left': []

},

'pain\_scale': {},

'symptoms': {}

}

return pain\_map

def add\_pain\_area(self, pain\_map, view, area\_data):

"""

Aggiunge area di dolore con intensità

"""

area = {

'id': generate\_id(),

'type': area\_data['type'], # acute, chronic, referred

'intensity': area\_data['intensity'], # 0-10

'quality': area\_data['quality'], # sharp, dull, burning, etc.

'coordinates': area\_data['coordinates'],

'shape': area\_data['shape'], # point, circle, polygon

'color': self.get\_pain\_color(area\_data['intensity']),

'pattern': area\_data.get('pattern', 'solid'), # solid, striped, dotted

'notes': area\_data.get('notes', '')

}

pain\_map['views'][view].append(area)

# Calcola statistiche

self.update\_pain\_statistics(pain\_map)

return area

def get\_pain\_color(self, intensity):

"""

Colore graduale basato su intensità dolore

"""

colors = {

0: '#FFFFFF', # No pain

1: '#FFF9E6', # Minimal

2: '#FFF3CC',

3: '#FFEB99',

4: '#FFD966',

5: '#FFC000', # Moderate

6: '#FF9900',

7: '#FF6600',

8: '#FF3300',

9: '#CC0000',

10: '#990000' # Severe

}

return colors.get(intensity, '#FF0000')

def generate\_dermatome\_overlay(self):

"""

Genera overlay dermatomeri per valutazione neurologica

"""

dermatomes = {

'C2': {'area': 'occipital\_region', 'color': '#E8F5E9'},

'C3': {'area': 'neck\_lateral', 'color': '#C8E6C9'},

'C4': {'area': 'shoulder\_top', 'color': '#A5D6A7'},

'C5': {'area': 'shoulder\_lateral', 'color': '#81C784'},

'C6': {'area': 'thumb\_index', 'color': '#66BB6A'},

'C7': {'area': 'middle\_finger', 'color': '#4CAF50'},

'C8': {'area': 'ring\_pinky', 'color': '#43A047'},

'T1': {'area': 'medial\_forearm', 'color': '#388E3C'},

# ... continua per tutti i dermatomeri

'L1': {'area': 'inguinal', 'color': '#E3F2FD'},

'L2': {'area': 'anterior\_thigh\_upper', 'color': '#BBDEFB'},

'L3': {'area': 'anterior\_thigh\_lower', 'color': '#90CAF9'},

'L4': {'area': 'medial\_leg', 'color': '#64B5F6'},

'L5': {'area': 'lateral\_leg\_dorsum\_foot', 'color': '#42A5F5'},

'S1': {'area': 'lateral\_foot\_heel', 'color': '#2196F3'},

'S2': {'area': 'posterior\_thigh', 'color': '#1E88E5'}

}

return dermatomes

**1.3 Integrazione 3D e Layer Anatomici**

class AnatomicalLayerSystem {

constructor() {

this.layers = new Map();

this.currentLayer = 'surface';

this.opacity = 1.0;

}

initializeLayers() {

// Layer superficie (pelle)

this.layers.set('surface', {

name: 'Cute e Tessuto Sottocutaneo',

visible: true,

opacity: 1.0,

elements: this.loadSurfaceElements(),

annotations: []

});

// Layer muscolare

this.layers.set('muscular', {

name: 'Sistema Muscolare',

visible: false,

opacity: 0.8,

elements: this.loadMuscularSystem(),

annotations: [],

subLayers: {

superficial: 'Muscoli Superficiali',

deep: 'Muscoli Profondi'

}

});

// Layer scheletrico

this.layers.set('skeletal', {

name: 'Sistema Scheletrico',

visible: false,

opacity: 0.9,

elements: this.loadSkeletalSystem(),

annotations: [],

highlights: {

joints: true,

vertebrae: true,

landmarks: true

}

});

// Layer nervoso

this.layers.set('nervous', {

name: 'Sistema Nervoso',

visible: false,

opacity: 0.7,

elements: this.loadNervousSystem(),

annotations: [],

subSystems: {

central: 'SNC',

peripheral: 'SNP',

autonomic: 'SNA'

}

});

// Layer vascolare

this.layers.set('vascular', {

name: 'Sistema Vascolare',

visible: false,

opacity: 0.6,

elements: this.loadVascularSystem(),

annotations: [],

types: {

arterial: { color: '#FF0000', visible: true },

venous: { color: '#0000FF', visible: true },

lymphatic: { color: '#00FF00', visible: false }

}

});

}

toggleLayer(layerName) {

const layer = this.layers.get(layerName);

if (layer) {

layer.visible = !layer.visible;

this.redraw();

}

}

addAnnotationToLayer(layerName, annotation) {

const layer = this.layers.get(layerName);

if (layer) {

annotation.id = generateId();

annotation.timestamp = new Date();

annotation.author = getCurrentUser();

layer.annotations.push(annotation);

this.saveAnnotation(annotation);

}

}

// Sistema di zoom intelligente

smartZoom(bodyPart) {

const zoomLevels = {

'full\_body': { scale: 1, center: { x: 50, y: 50 } },

'head\_neck': { scale: 3, center: { x: 50, y: 15 } },

'shoulder': { scale: 4, center: { x: 35, y: 30 } },

'spine': { scale: 2.5, center: { x: 50, y: 45 } },

'hand': { scale: 5, center: { x: 30, y: 58 } },

'knee': { scale: 4, center: { x: 45, y: 75 } },

'foot': { scale: 5, center: { x: 42, y: 90 } }

};

const zoom = zoomLevels[bodyPart] || zoomLevels['full\_body'];

this.animateZoom(zoom.scale, zoom.center);

}

}

**2. INTERFACCIA UTENTE BODY MAPPING**

**2.1 Editor Interattivo**

class BodyMapEditor {

constructor(container) {

this.container = container;

this.canvas = this.createCanvas();

this.ctx = this.canvas.getContext('2d');

this.currentTool = 'marker';

this.currentMarkerType = 'pain';

this.currentView = 'anterior';

this.isDrawing = false;

this.history = [];

this.historyStep = -1;

}

// Strumenti di marcatura

tools = {

marker: {

name: 'Marcatore Punto',

icon: '📍',

cursor: 'crosshair',

action: (x, y) => {

this.addMarker(x, y, this.currentMarkerType);

}

},

area: {

name: 'Area di Trattamento',

icon: '⭕',

cursor: 'crosshair',

action: (startX, startY, endX, endY) => {

this.drawArea(startX, startY, endX, endY);

}

},

freehand: {

name: 'Disegno Libero',

icon: '✏️',

cursor: 'crosshair',

action: (path) => {

this.drawFreehand(path);

}

},

text: {

name: 'Annotazione Testo',

icon: '💬',

cursor: 'text',

action: (x, y, text) => {

this.addTextAnnotation(x, y, text);

}

},

measure: {

name: 'Misura Distanza',

icon: '📏',

cursor: 'crosshair',

action: (startX, startY, endX, endY) => {

this.measureDistance(startX, startY, endX, endY);

}

},

eraser: {

name: 'Gomma',

icon: '🗑️',

cursor: 'grab',

action: (x, y) => {

this.eraseAt(x, y);

}

}

};

// Preset comuni per patologie

presets = {

lombalgia: {

name: 'Lombalgia',

markers: [

{ x: 50, y: 48, type: 'pain', view: 'posterior', notes: 'Dolore lombare L4-L5' },

{ x: 48, y: 50, type: 'trigger\_point', view: 'posterior', notes: 'TP quadrato dei lombi' },

{ x: 52, y: 50, type: 'trigger\_point', view: 'posterior', notes: 'TP quadrato dei lombi' }

]

},

cervicalgia: {

name: 'Cervicalgia',

markers: [

{ x: 50, y: 23, type: 'pain', view: 'posterior', notes: 'Dolore cervicale C5-C6' },

{ x: 45, y: 25, type: 'trigger\_point', view: 'posterior', notes: 'TP trapezio superiore' },

{ x: 55, y: 25, type: 'trigger\_point', view: 'posterior', notes: 'TP trapezio superiore' }

]

},

sciatica: {

name: 'Sciatalgia',

markers: [

{ x: 50, y: 50, type: 'pain', view: 'posterior', notes: 'Origine L5-S1' },

{ x: 48, y: 60, type: 'radiating\_pain', view: 'posterior', notes: 'Irradiazione gluteo' },

{ x: 45, y: 70, type: 'radiating\_pain', view: 'posterior', notes: 'Irradiazione coscia' },

{ x: 43, y: 85, type: 'numbness', view: 'posterior', notes: 'Parestesia gamba' }

]

}

};

// Salvataggio e caricamento

saveToPatientRecord() {

const mapData = {

patientId: this.patientId,

sessionId: this.sessionId,

date: new Date(),

therapist: getCurrentUser(),

views: {},

markers: this.markers,

drawings: this.drawings,

annotations: this.annotations,

measurements: this.measurements

};

// Salva screenshot di ogni vista

['anterior', 'posterior', 'lateral\_dx', 'lateral\_sx'].forEach(view => {

mapData.views[view] = this.captureView(view);

});

// Salva nel database

return saveBodyMap(mapData);

}

loadFromPatientRecord(recordId) {

const mapData = loadBodyMap(recordId);

this.markers = mapData.markers || [];

this.drawings = mapData.drawings || [];

this.annotations = mapData.annotations || [];

this.measurements = mapData.measurements || [];

this.redraw();

}

// Esportazione

exportToPDF() {

const pdf = new jsPDF();

// Aggiungi titolo

pdf.setFontSize(16);

pdf.text('Mappa Zone Trattamento', 105, 20, { align: 'center' });

// Info paziente

pdf.setFontSize(10);

pdf.text(`Paziente: ${this.patientName}`, 20, 30);

pdf.text(`Data: ${new Date().toLocaleDateString('it-IT')}`, 20, 35);

pdf.text(`Terapista: ${getCurrentUser().name}`, 20, 40);

// Aggiungi immagini delle viste

let yPosition = 50;

['anterior', 'posterior'].forEach(view => {

const imgData = this.captureView(view);

pdf.addImage(imgData, 'PNG', 20, yPosition, 80, 100);

pdf.text(view === 'anterior' ? 'Vista Anteriore' : 'Vista Posteriore', 60, yPosition - 5, { align: 'center' });

yPosition += 110;

});

// Legenda

pdf.addPage();

pdf.setFontSize(12);

pdf.text('Legenda Marcatori:', 20, 20);

let legendY = 30;

Object.entries(this.markerTypes).forEach(([key, marker]) => {

pdf.setFillColor(marker.color);

pdf.circle(25, legendY, 2, 'F');

pdf.text(`${marker.name}: ${marker.description}`, 35, legendY + 1);

legendY += 8;

});

// Note

if (this.annotations.length > 0) {

pdf.text('Note:', 20, legendY + 10);

let noteY = legendY + 20;

this.annotations.forEach(ann => {

pdf.text(`• ${ann.text}`, 25, noteY);

noteY += 6;

});

}

return pdf;

}

}

**3. INTEGRAZIONE CON SISTEMA CLINICO**

**3.1 Collegamento con Valutazioni**

class ClinicalIntegration:

def link\_to\_assessment(self, body\_map, assessment\_id):

"""

Collega mappa corporea a valutazione clinica

"""

assessment = self.get\_assessment(assessment\_id)

# Correlazione automatica dolore-test

correlations = []

for marker in body\_map['markers']:

if marker['type'] == 'pain':

# Suggerisci test basati sulla zona

suggested\_tests = self.suggest\_tests\_for\_area(

marker['anatomicalLocation']

)

correlations.append({

'marker': marker,

'tests': suggested\_tests,

'positive\_tests': [] # Da compilare dopo esecuzione

})

# Genera report integrato

integrated\_report = {

'visual\_documentation': body\_map,

'clinical\_findings': assessment,

'correlations': correlations,

'treatment\_targets': self.identify\_treatment\_targets(body\_map)

}

return integrated\_report

def track\_progress(self, patient\_id):

"""

Confronta mappe nel tempo per valutare progressi

"""

maps = self.get\_all\_body\_maps(patient\_id)

if len(maps) < 2:

return None

progress = {

'pain\_reduction': self.calculate\_pain\_reduction(maps),

'area\_reduction': self.calculate\_area\_reduction(maps),

'new\_areas': self.identify\_new\_areas(maps),

'resolved\_areas': self.identify\_resolved\_areas(maps),

'timeline': self.create\_progress\_timeline(maps)

}

# Genera visualizzazione comparativa

comparison\_image = self.generate\_comparison\_image(

maps[0], # Prima mappa

maps[-1] # Ultima mappa

)

progress['visual\_comparison'] = comparison\_image

return progress

**3.2 Template per Patologie Comuni**

const pathologyTemplates = {

// Template per diverse condizioni

shoulder\_impingement: {

name: 'Sindrome da Impingement Spalla',

defaultView: 'anterior',

commonMarkers: [

{

location: 'acromion',

type: 'pain',

description: 'Dolore sottoacromiale'

},

{

location: 'anterior\_shoulder',

type: 'inflammation',

description: 'Infiammazione borsa subacromiale'

},

{

location: 'lateral\_arm',

type: 'radiating\_pain',

description: 'Possibile irradiazione laterale'

}

],

instructions: 'Marcare punto di massimo dolore durante elevazione'

},

plantar\_fasciitis: {

name: 'Fascite Plantare',

defaultView: 'plantar',

commonMarkers: [

{

location: 'medial\_calcaneus',

type: 'pain',

description: 'Dolore inserzione calcaneare'

},

{

location: 'plantar\_arch',

type: 'treatment\_area',

description: 'Area di trattamento fascia'

}

],

instructions: 'Identificare punto di massima dolorabilità al risveglio'

},

carpal\_tunnel: {

name: 'Sindrome Tunnel Carpale',

defaultView: 'palmar',

commonMarkers: [

{

location: 'carpal\_tunnel',

type: 'compression',

description: 'Compressione nervo mediano'

},

{

location: 'thumb\_to\_ring',

type: 'numbness',

description: 'Parestesie territorio mediano'

}

],

instructions: 'Mappare distribuzione parestesie notturne'

}

};

// Funzione per applicare template

function applyPathologyTemplate(templateName) {

const template = pathologyTemplates[templateName];

if (template) {

// Cambia vista

bodyMapEditor.setView(template.defaultView);

// Aggiungi marcatori comuni

template.commonMarkers.forEach(marker => {

const coords = getAnatomicalCoordinates(marker.location);

bodyMapEditor.addMarker(

coords.x,

coords.y,

marker.type,

template.defaultView,

marker.description

);

});

// Mostra istruzioni

showInstructions(template.instructions);

}

}

**4. ANALISI E STATISTICHE**

**4.1 Heat Map del Dolore**

class PainHeatMapAnalyzer:

def generate\_clinic\_heatmap(self, time\_period):

"""

Genera heat map delle zone più trattate nella clinica

"""

all\_maps = self.get\_all\_body\_maps\_in\_period(time\_period)

# Crea griglia per heat map

heatmap\_grid = np.zeros((100, 100)) # 100x100 pixel grid

for body\_map in all\_maps:

for marker in body\_map['markers']:

if marker['type'] in ['pain', 'treatment\_area']:

# Aggiungi peso alla griglia

x, y = int(marker['x']), int(marker['y'])

radius = 5

intensity = marker.get('intensity', 5)

# Applica gaussian blur per smooth heat map

for i in range(-radius, radius+1):

for j in range(-radius, radius+1):

if 0 <= x+i < 100 and 0 <= y+j < 100:

distance = np.sqrt(i\*\*2 + j\*\*2)

if distance <= radius:

weight = intensity \* np.exp(-distance\*\*2 / (2 \* (radius/2)\*\*2))

heatmap\_grid[y+j, x+i] += weight

# Normalizza

heatmap\_grid = heatmap\_grid / np.max(heatmap\_grid) if np.max(heatmap\_grid) > 0 else heatmap\_grid

# Genera immagine heat map

plt.figure(figsize=(10, 12))

plt.imshow(heatmap\_grid, cmap='hot', interpolation='gaussian')

plt.colorbar(label='Frequenza Trattamento')

plt.title(f'Heat Map Zone Trattamento - {time\_period}')

# Identifica hot spots

hotspots = self.identify\_hotspots(heatmap\_grid)

return {

'heatmap': heatmap\_grid,

'hotspots': hotspots,

'statistics': self.calculate\_statistics(all\_maps)

}

def identify\_hotspots(self, heatmap):

"""

Identifica le zone più frequentemente trattate

"""

threshold = np.percentile(heatmap, 90) # Top 10% delle zone

hotspots = []

# Find contours of hot areas

hot\_areas = heatmap > threshold

labeled, num\_features = label(hot\_areas)

for i in range(1, num\_features + 1):

area\_mask = labeled == i

center = center\_of\_mass(area\_mask)

# Identifica zona anatomica

anatomical\_zone = self.get\_anatomical\_zone(center[1], center[0])

hotspots.append({

'center': center,

'anatomical\_zone': anatomical\_zone,

'frequency': np.sum(area\_mask),

'max\_intensity': np.max(heatmap[area\_mask])

})

return sorted(hotspots, key=lambda x: x['frequency'], reverse=True)

**5. MOBILE INTEGRATION**

**5.1 Touch Interface per Tablet**

class TouchBodyMapper {

constructor(canvas) {

this.canvas = canvas;

this.touches = new Map();

this.gestures = new GestureRecognizer();

this.initTouchEvents();

}

initTouchEvents() {

// Touch start

this.canvas.addEventListener('touchstart', (e) => {

e.preventDefault();

const touches = e.changedTouches;

for (let touch of touches) {

const point = this.getTouchPoint(touch);

this.touches.set(touch.identifier, {

start: point,

current: point,

tool: this.currentTool

});

if (this.currentTool === 'marker') {

this.showMarkerPreview(point);

}

}

});

// Touch move

this.canvas.addEventListener('touchmove', (e) => {

e.preventDefault();

const touches = e.changedTouches;

for (let touch of touches) {

const point = this.getTouchPoint(touch);

const touchData = this.touches.get(touch.identifier);

if (touchData) {

touchData.current = point;

if (this.currentTool === 'freehand') {

this.drawLine(touchData.start, point);

touchData.start = point;

} else if (this.currentTool === 'area') {

this.updateAreaPreview(touchData.start, point);

}

}

}

// Gesture recognition

if (e.touches.length === 2) {

const gesture = this.gestures.recognize(e.touches);

if (gesture === 'pinch') {

this.handlePinchZoom(e.touches);

} else if (gesture === 'rotate') {

this.handleRotation(e.touches);

}

}

});

// Touch end

this.canvas.addEventListener('touchend', (e) => {

e.preventDefault();

const touches = e.changedTouches;

for (let touch of touches) {

const touchData = this.touches.get(touch.identifier);

if (touchData) {

if (this.currentTool === 'marker') {

this.placeMarker(touchData.current);

} else if (this.currentTool === 'area') {

this.createArea(touchData.start, touchData.current);

}

this.touches.delete(touch.identifier);

}

}

});

}

// Quick action menu

showQuickMenu(x, y) {

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

menu.className = 'quick-menu';

menu.style.left = x + 'px';

menu.style.top = y + 'px';

const actions = [

{ icon: '✕', action: 'pain', label: 'Dolore' },

{ icon: '●', action: 'trigger', label: 'Trigger Point' },

{ icon: '○', action: 'treatment', label: 'Trattamento' },

{ icon: '💬', action: 'note', label: 'Nota' }

];

actions.forEach(item => {

const button = document.createElement('button');

button.innerHTML = `${item.icon} ${item.label}`;

button.onclick = () => {

this.performQuickAction(item.action, x, y);

menu.remove();

};

menu.appendChild(button);

});

document.body.appendChild(menu);

}

}

**6. STAMPA E DOCUMENTAZIONE**

**6.1 Report con Body Maps**

def generate\_treatment\_report\_with\_maps(patient\_id, session\_id):

"""

Genera report con mappe corporee integrate

"""

from reportlab.lib.pagesizes import A4

from reportlab.platypus import SimpleDocTemplate, Paragraph, Image, Table

# Crea documento

doc = SimpleDocTemplate(f"report\_{patient\_id}\_{session\_id}.pdf", pagesize=A4)

story = []

# Intestazione

story.append(Paragraph("REFERTO VALUTAZIONE FISIOTERAPICA", title\_style))

# Info paziente

patient\_info = get\_patient\_info(patient\_id)

story.append(Paragraph(f"Paziente: {patient\_info['name']}", normal\_style))

story.append(Paragraph(f"Data: {datetime.now().strftime('%d/%m/%Y')}", normal\_style))

# Mappa corporea

story.append(Paragraph("LOCALIZZAZIONE SINTOMATOLOGIA", heading\_style))

# Genera immagini delle mappe

body\_maps = get\_body\_maps(session\_id)

# Crea tabella con viste

map\_data = []

for view in ['anterior', 'posterior']:

if view in body\_maps:

img = Image(body\_maps[view], width=200, height=250)

map\_data.append([

view.capitalize(),

img

])

if map\_data:

map\_table = Table(map\_data)

story.append(map\_table)

# Legenda

story.append(Paragraph("Legenda:", small\_style))

legend\_data = [

['✕', 'Punto di dolore principale'],

['●', 'Trigger point'],

['○', 'Area di trattamento'],

['▲', 'Infiammazione']

]

legend\_table = Table(legend\_data)

story.append(legend\_table)

# Note cliniche

clinical\_notes = get\_clinical\_notes(session\_id)

if clinical\_notes:

story.append(Paragraph("NOTE CLINICHE", heading\_style))

for note in clinical\_notes:

story.append(Paragraph(f"• {note}", normal\_style))

# Genera PDF

doc.build(story)

return f"report\_{patient\_id}\_{session\_id}.pdf"

**7. DATABASE SCHEMA**

-- Tabella per memorizzare le mappe corporee

CREATE TABLE body\_maps (

id SERIAL PRIMARY KEY,

patient\_id INTEGER REFERENCES patients(id),

session\_id INTEGER REFERENCES therapy\_sessions(id),

created\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

therapist\_id INTEGER REFERENCES users(id),

map\_type VARCHAR(50), -- 'pain\_map', 'treatment\_map', 'progress\_map'

status VARCHAR(20) DEFAULT 'draft'

);

-- Tabella per i marcatori

CREATE TABLE body\_map\_markers (

id SERIAL PRIMARY KEY,

body\_map\_id INTEGER REFERENCES body\_maps(id),

marker\_type VARCHAR(50), -- 'pain', 'trigger\_point', 'treatment\_area', etc.

view\_type VARCHAR(20), -- 'anterior', 'posterior', 'lateral\_right', 'lateral\_left'

x\_coordinate DECIMAL(5,2),

y\_coordinate DECIMAL(5,2),

intensity INTEGER CHECK (intensity >= 0 AND intensity <= 10),

size VARCHAR(20), -- 'point', 'small', 'medium', 'large'

color VARCHAR(7), -- Hex color

notes TEXT,

anatomical\_region VARCHAR(100),

anatomical\_structure VARCHAR(100),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Tabella per disegni e aree

CREATE TABLE body\_map\_drawings (

id SERIAL PRIMARY KEY,

body\_map\_id INTEGER REFERENCES body\_maps(id),

drawing\_type VARCHAR(50), -- 'area', 'line', 'polygon', 'freehand'

view\_type VARCHAR(20),

path\_data TEXT, -- SVG path o coordinate JSON

fill\_color VARCHAR(7),

stroke\_color VARCHAR(7),

opacity DECIMAL(3,2),

notes TEXT

);

-- Tabella per annotazioni testuali

CREATE TABLE body\_map\_annotations (

id SERIAL PRIMARY KEY,

body\_map\_id INTEGER REFERENCES body\_maps(id),

view\_type VARCHAR(20),

x\_coordinate DECIMAL(5,2),

y\_coordinate DECIMAL(5,2),

text\_content TEXT,

font\_size INTEGER DEFAULT 12,

text\_color VARCHAR(7) DEFAULT '#000000'

);

-- Tabella per misurazioni

CREATE TABLE body\_map\_measurements (

id SERIAL PRIMARY KEY,

body\_map\_id INTEGER REFERENCES body\_maps(id),

measurement\_type VARCHAR(50), -- 'distance', 'angle', 'circumference'

start\_x DECIMAL(5,2),

start\_y DECIMAL(5,2),

end\_x DECIMAL(5,2),

end\_y DECIMAL(5,2),

value DECIMAL(10,2),

unit VARCHAR(10), -- 'cm', 'mm', 'degrees'

notes TEXT

);

-- Vista per analisi statistiche

CREATE VIEW body\_map\_statistics AS

SELECT

bm.patient\_id,

bm.created\_date,

bmm.anatomical\_region,

bmm.marker\_type,

COUNT(\*) as marker\_count,

AVG(bmm.intensity) as avg\_intensity

FROM body\_maps bm

JOIN body\_map\_markers bmm ON bm.id = bmm.body\_map\_id

GROUP BY bm.patient\_id, bm.created\_date, bmm.anatomical\_region, bmm.marker\_type;