**Repo: https://github.com/rafadagalera/SkillPlus2030**

**Feito por:**

Rafael Nascimento rm5531117
Isabelle Torricelli rm552806

## Solução Proposta

O **SkillPlus2030** foi desenvolvido para abordar o desafio de capacitação profissional através de uma solução mobile completa e personalizada. A aplicação implementa uma arquitetura modular que combina conceitos de orientação a objetos, separação de responsabilidades e organização estrutural.

### Arquitetura e Organização

A solução adota uma arquitetura em camadas com separação clara de responsabilidades:

## Estrutura do Projeto

```
src/
├── components/       # Componentes reutilizáveis
├── navigation/       # Configuração de navegação
├── screens/          # Telas da aplicação
│  ├──Auth/           # Login e Registro
│  └──Home/           # Trilhas, Autoavaliação, Progresso
├── theme/            # Estilos e temas
└── utils/            # Funções utilitárias e lógica de negócio
```

- **Camada de Apresentação (`screens/`)**: Componentes funcionais React que representam as telas da aplicação, utilizando hooks para gerenciamento de estado local
- **Camada de Componentes (`components/`)**: Componentes reutilizáveis como `TrailCard`
- **Camada de Navegação (`navigation/`)**: Configuração centralizada usando React Navigation
- **Camada de Utilitários (`utils/`)**: Lógica de negócio e serviços de persistência de dados
- **Camada de Estilização (`theme/`)**: Sistema de design unificado com constantes de cores, espaçamento e tipografia

## Funcionalidades Principais

- **Autoavaliação de Competências**: Avalie seu nível atual em diferentes habilidades (Comunicação, Pensamento Crítico, IA Básica, Sustentabilidade, Trabalho em Equipe, Gestão do Tempo)
- **Trilhas de Aprendizado**: Acesse trilhas estruturadas com conteúdos em vídeo, texto e quizzes
- **Recomendações Personalizadas**: Receba sugestões de trilhas baseadas nas suas autoavaliações
- **Acompanhamento de Progresso**: Monitore seu desenvolvimento e conquistas
- **Sistema de Gamificação**: Ganhe pontos e badges ao completar atividades

## Implementação da Solução

- **Sistema de Autoavaliação e Recomendações**

O módulo de recomendações (`utils/recommendations.ts`) implementa um algoritmo inteligente que:

- **Interface `Trail`**: Define a estrutura de dados de uma trilha de aprendizado com propriedades tipadas (id, title, description, duration, level, category, skills, lessons)

- **Interface `Lesson`**: Especifica os tipos de conteúdo (video, text, quiz) com duração e status de conclusão
- **Função `getRecommendedTrails()`**: Analisa as autoavaliações do usuário, identifica competências com notas abaixo de 7 e retorna trilhas que desenvolvem essas habilidades, ordenadas por relevância

- **Persistência de Dados com AsyncStorage**

O módulo `storage.ts` implementa um serviço de persistência seguindo padrões de encapsulamento:

- **Interfaces TypeScript**: `Profile`, `Assessment`, `TrailProgress`, `UserProgress` - definem contratos claros para os dados
- **Funções especializadas**: Cada função tem responsabilidade única (saveAssessment, getAssessments, saveTrailProgress, etc.)
- **Abstração de armazenamento**: O AsyncStorage é encapsulado, permitindo futura migração para banco de dados sem impactar o restante da aplicação

- **Sistema de Gamificação**

Implementação de um sistema de pontos e badges que:

- **Função `addPoints()`**: Adiciona pontos de forma incremental e persistente
- **Função `checkBadges()`**: Verifica condições para desbloqueio de badges (Primeiros Passos, Trilha Completa, Avaliador, Especialista)
- **Função `getUserProgress()`**: Agrega dados de múltiplas fontes para gerar um resumo completo do progresso do usuário

- **Navegação Hierárquica**

A navegação utiliza múltiplos níveis de stack:

- **RootStack**: Gerencia autenticação vs. aplicação principal
- **AuthStack**: Login e registro
- **DrawerNavigator**: Menu lateral para acesso a perfil e configurações
- **TabNavigator**: Navegação por abas na área principal

- **Componentes Reutilizáveis**

O componente `TrailCard` demonstra:

- **Props tipadas**: Interface `Props` define o contrato do componente
- **Composição**: Utiliza componentes nativos do React Native de forma combinada
- **Reutilização**: Pode ser usado em diferentes contextos

## Conceitos de Orientação a Objetos Aplicados

Asolução incorpora princípios fundamentais de O.O.:

- 1. Encapsulamento: Dados e funções relacionadas são agrupadas em módulos (`storage.ts`, `recommendations.ts`)

- 2. Abstração**: Interfaces TypeScript definem contratos sem expor implementação
- 3. **Separação de Responsabilidades**: Cada módulo tem uma função específica e bem definida

- 4. **Reutilização**: Componentes e funções utilitárias são projetados para serem reutilizáveis
- 5. **Type Safety**: TypeScript garante consistência de tipos em toda a aplicação

6.

## Resultados Esperados

Com a implementação do **SkillPlus2030**, espera-se alcançar os seguintes resultados:

- 1. **Aumento da Autoconsciência Profissional**
- 2. **Desenvolvimento Personalizado de Competências**
- 3. **Engajamento Sustentado**
- 4. **Acesso Democratizado ao Aprendizado**
- 5. **Mensuração de Progresso**

## Impacto Positivo Almejado

### Impacto Individual

•**Desenvolvimento Profissional Contínuo**: Profissionais terão acesso a uma ferramenta que facilita o aprendizado contínuo e o desenvolvimento de competências essenciais para 2030 - **Preparação para o Futuro do Trabalho**: Foco em competências como IA, sustentabilidade e soft skills prepara usuários para as demandas do mercado futuro - **Autonomia no Aprendizado**: Sistema de recomendações personalizadas permite aprendizado autodirigido e eficiente

### Impacto Organizacional

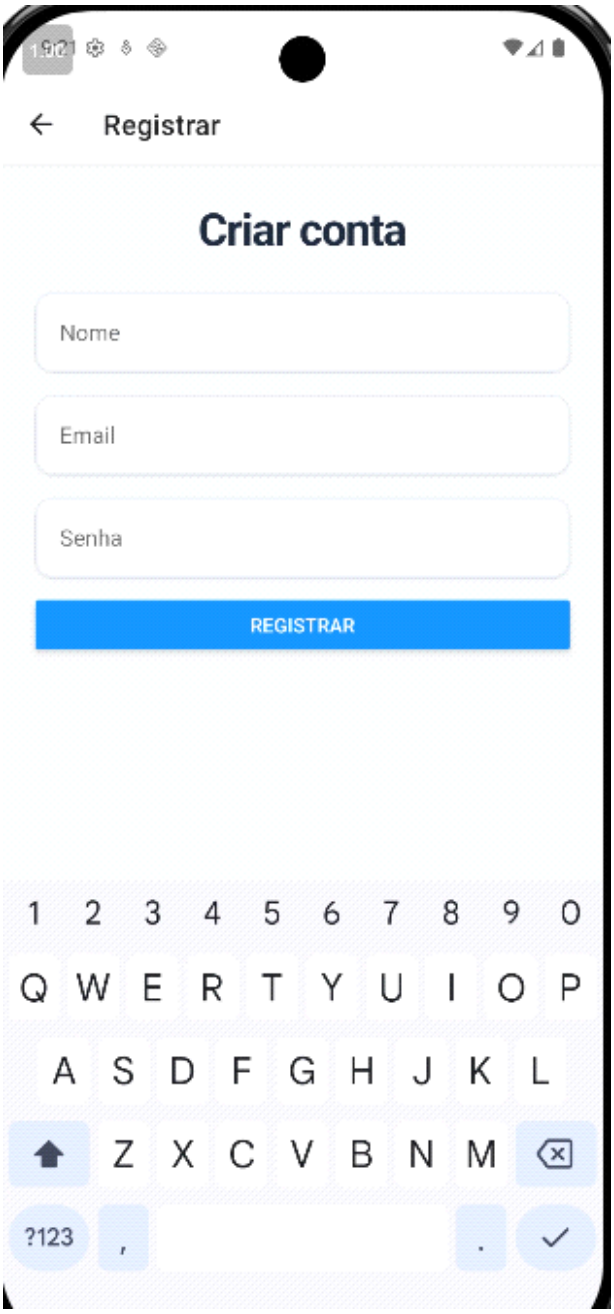
•**Capacitação de Equipes**: Organizações podem utilizar a plataforma para capacitar equipes de forma escalável - **Redução de Custos de Treinamento**: Solução mobile reduz necessidade de treinamentos presenciais e infraestrutura física - **Mensuração de Desenvolvimento**: Métricas de progresso permitem acompanhamento do desenvolvimento de competências em nível organizacional

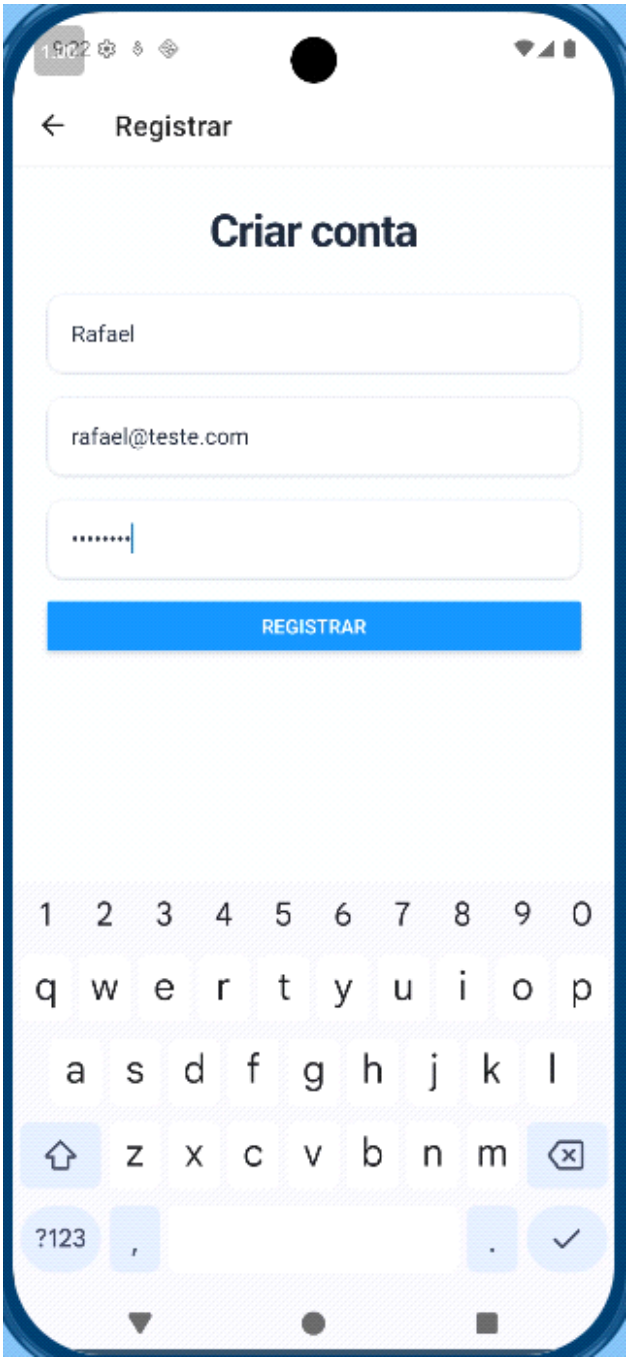
### Impacto Social

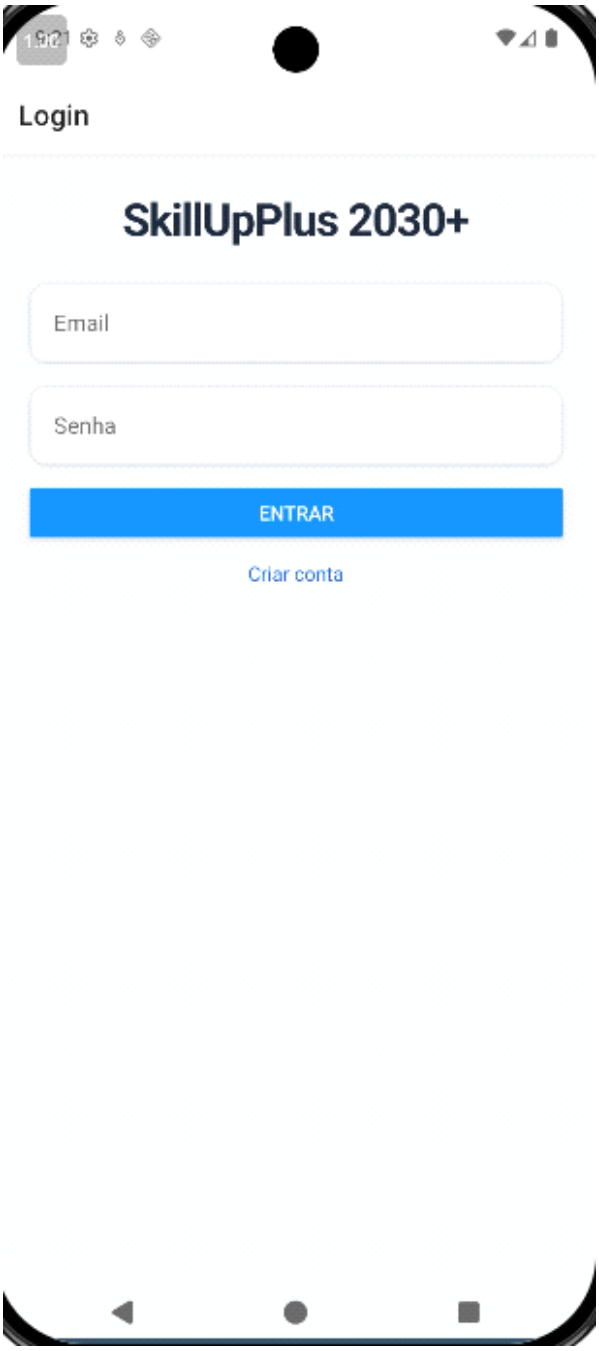
•**Democratização da Educação**: Acesso facilitado a conteúdos de qualidade através de dispositivo mobile - **Preparação para Transformação Digital**: Contribuição para preparação da força de trabalho para mudanças tecnológicas - **Sustentabilidade**: Promoção de competências relacionadas à sustentabilidade no ambiente profissional

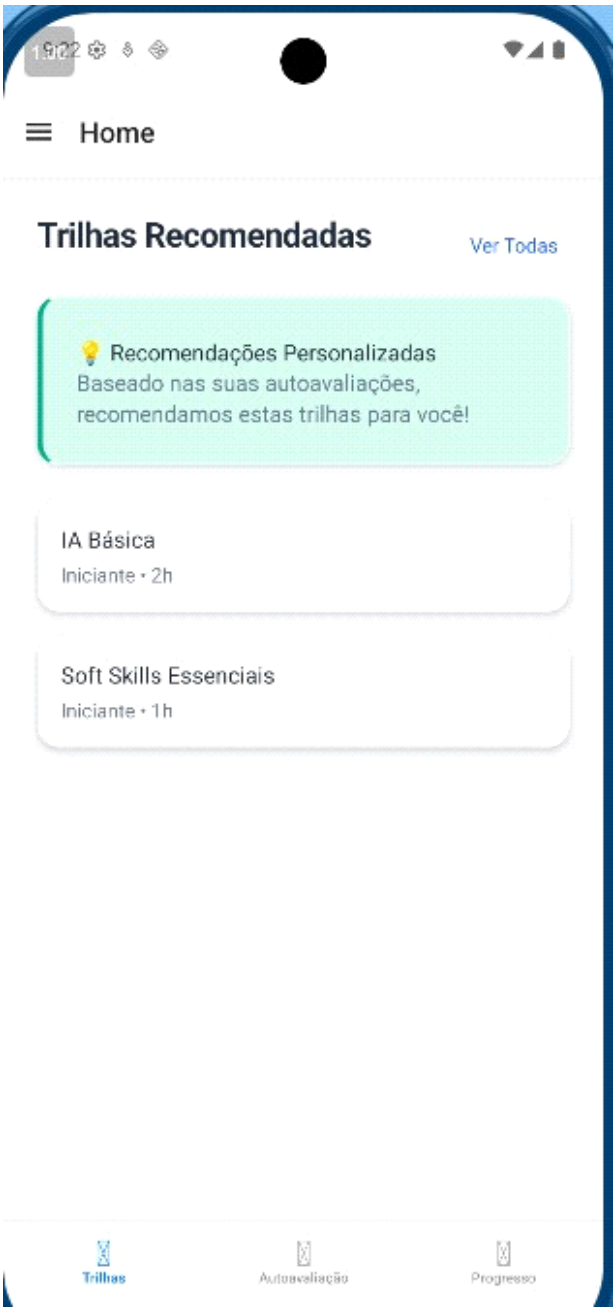
O **SkillPlus2030** representa uma solução completa e inovadora para o desafio de capacitação profissional, combinando tecnologia moderna, design centrado no usuário e uma abordagem baseada em dados para personalização do aprendizado.

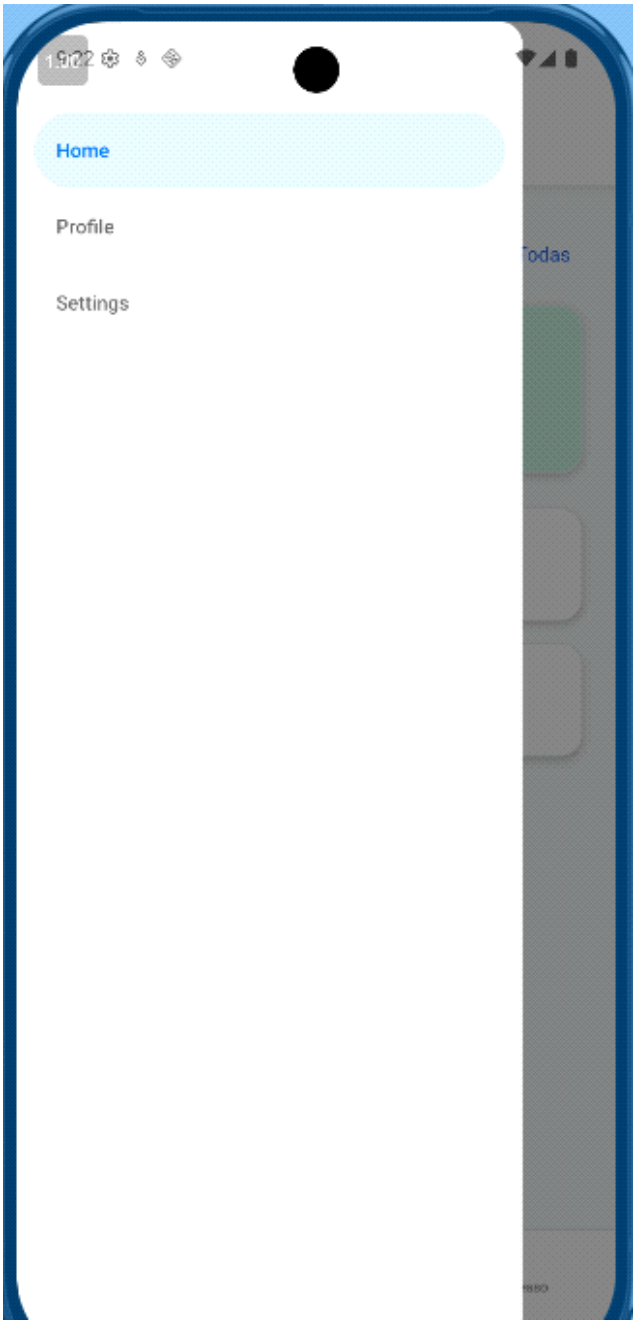
## Prints da aplicação:

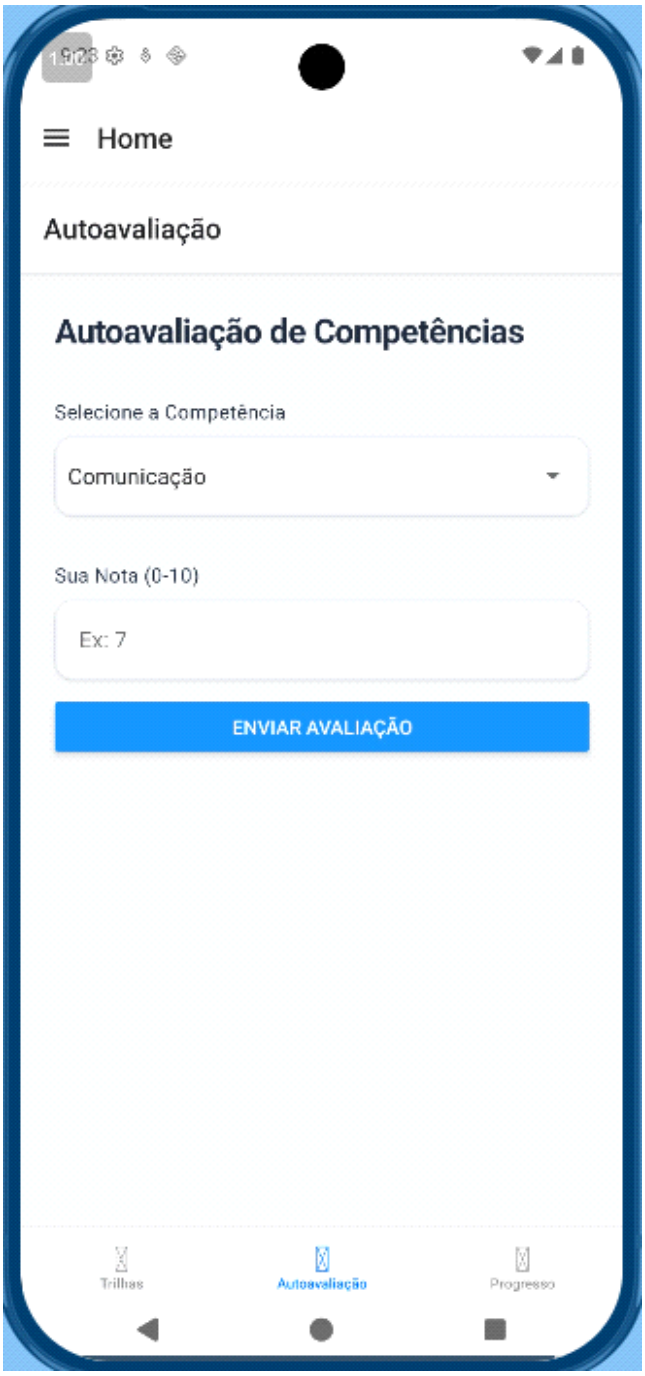


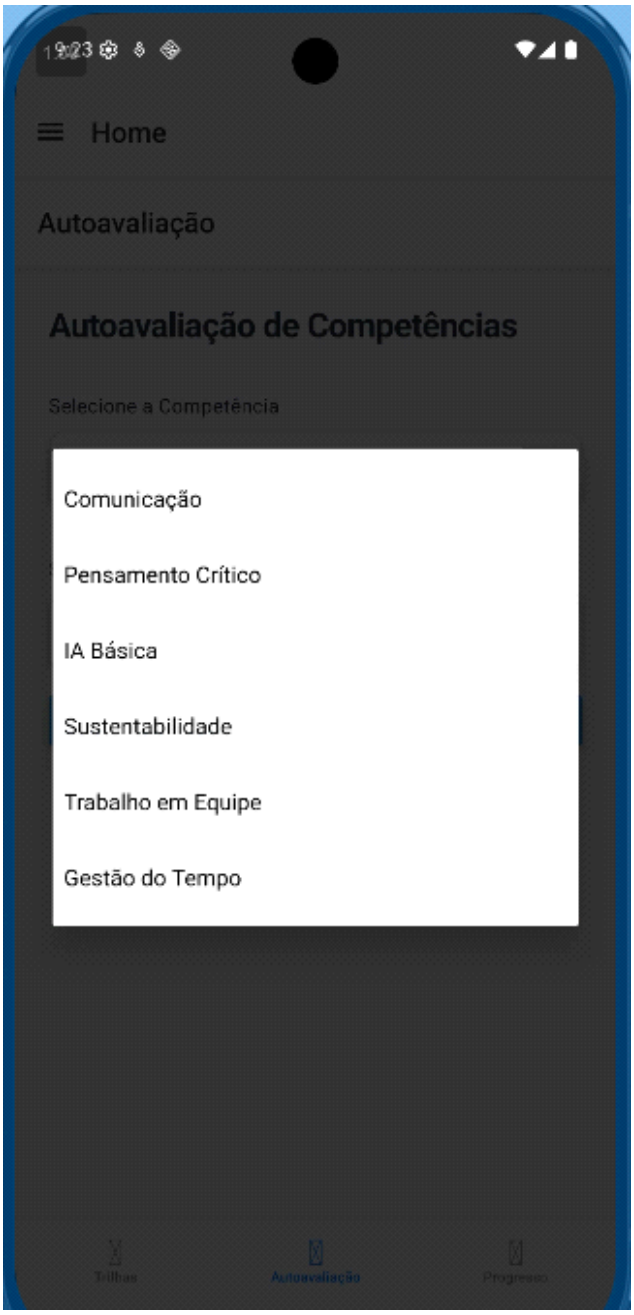

Registrar
Criar conta
Rafael
rafael@teste.com
REGISTRAR

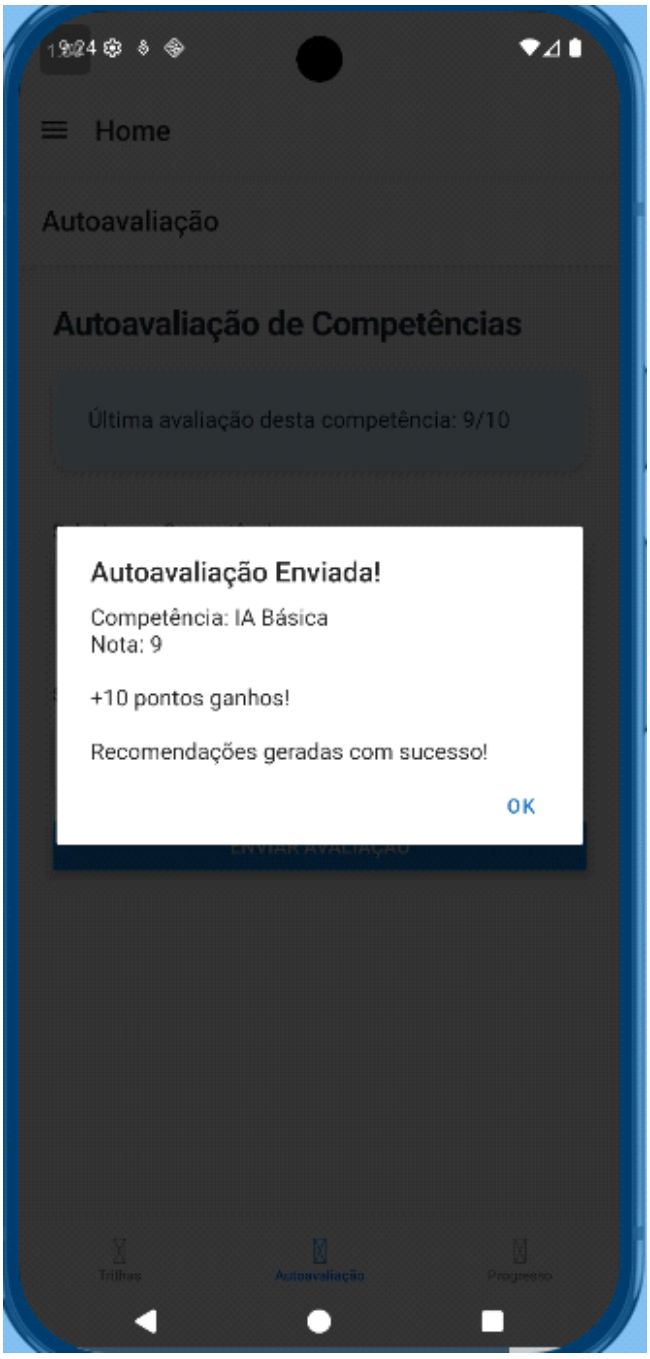

Login
SkillUpPlus 2030+
Email
Senha
ENTRAR
Criar conta

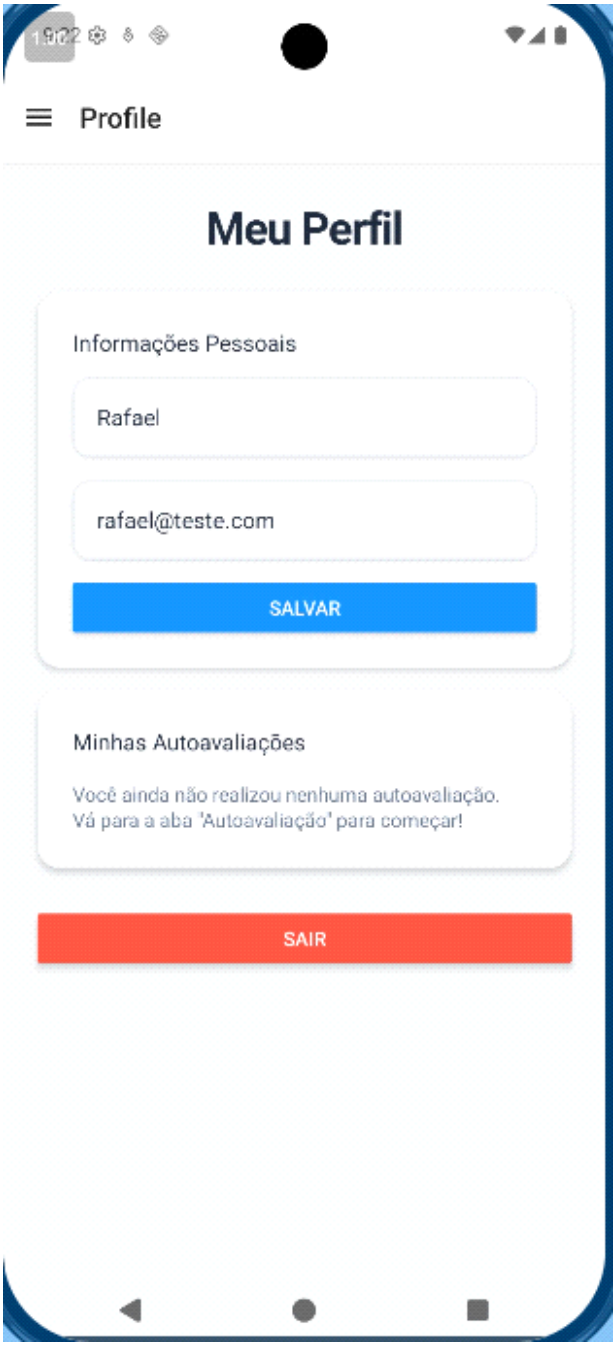

Home
Trilhas Recomendadas
Ver Todas
Recomendações Personalizadas
Baseado nas suas autoavaliações, recomendamos estas trilhas para você!
IA Básica
Iniciante • 2h
Soft Skills Essenciais
Iniciante • 1h
Trilhas
Autoavaliação
Progresso

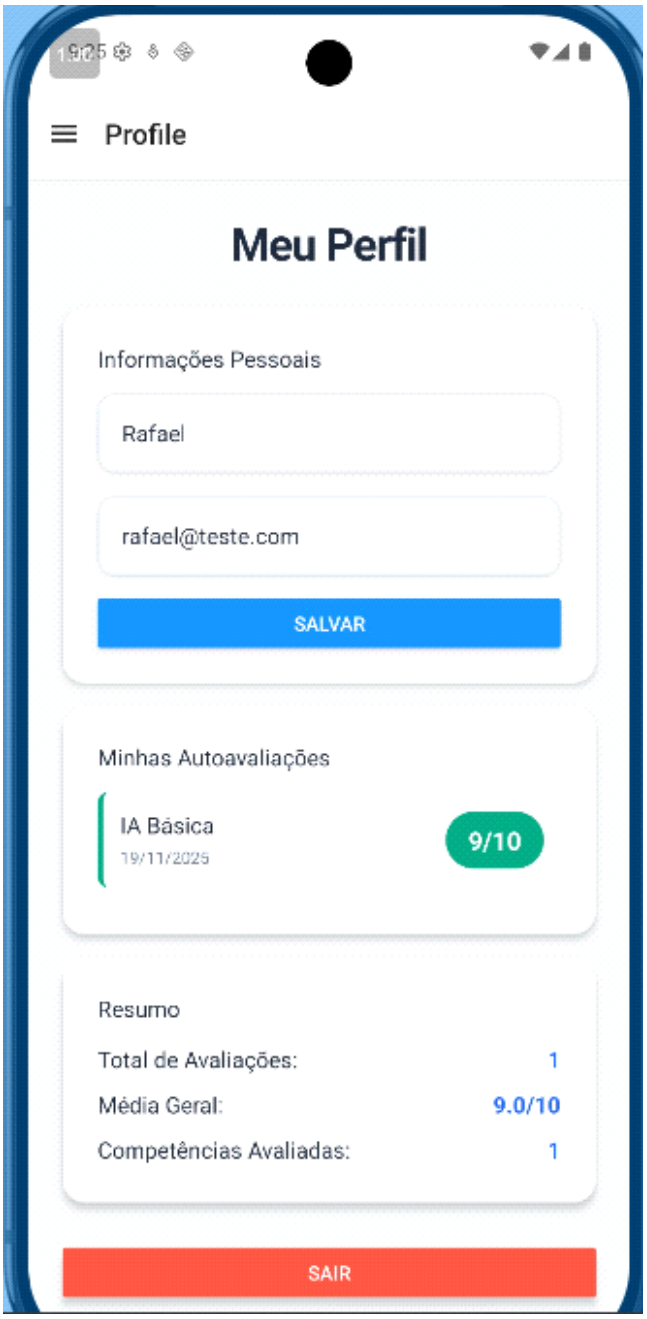

Home
Profile
Settings
Todas

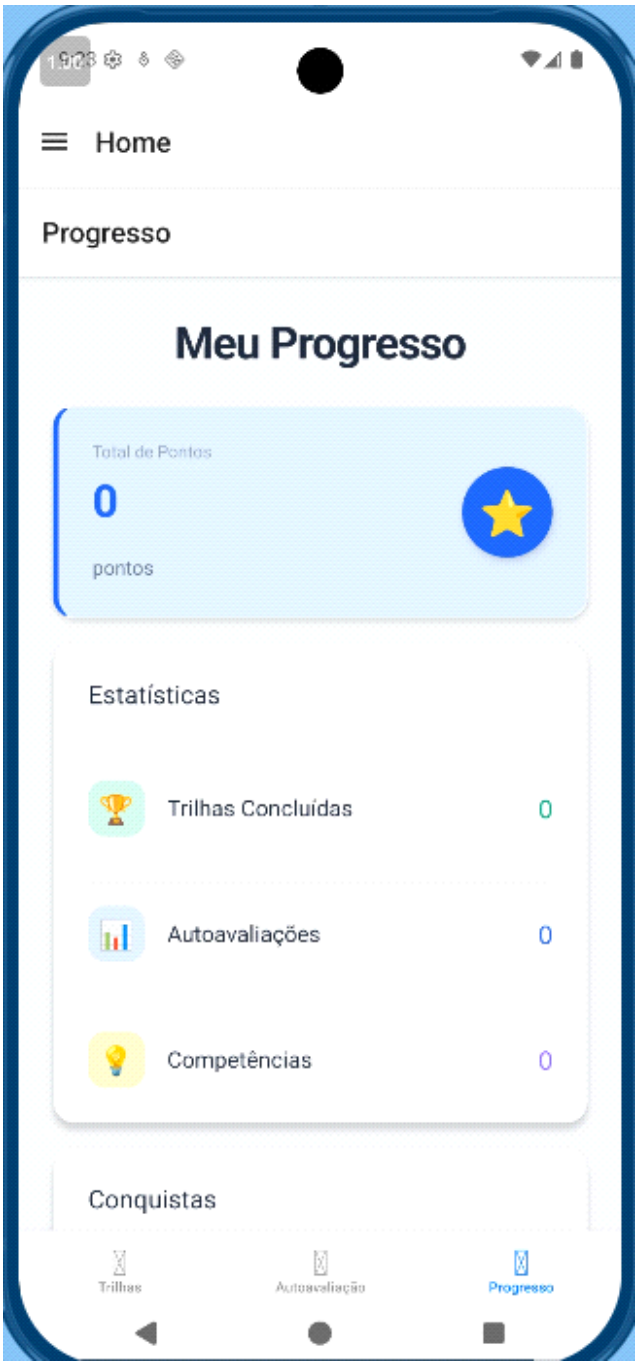

Home
Autoavaliação
Autoavaliação de Competências
Selecione a Competência
Comunicação
Sua Nota (0-10)
Ex: 7
ENVIAR AVALIAÇÃO
Trilhas
Autoavaliação
Progresso


19:23
Home
Autoavaliação
Autoavaliação de Competências
Selecione a Competência
Comunicação
Pensamento Crítico
IA Básica
Sustentabilidade
Trabalho em Equipe
Gestão do Tempo
Trilhas
Autoavaliação
Progresso

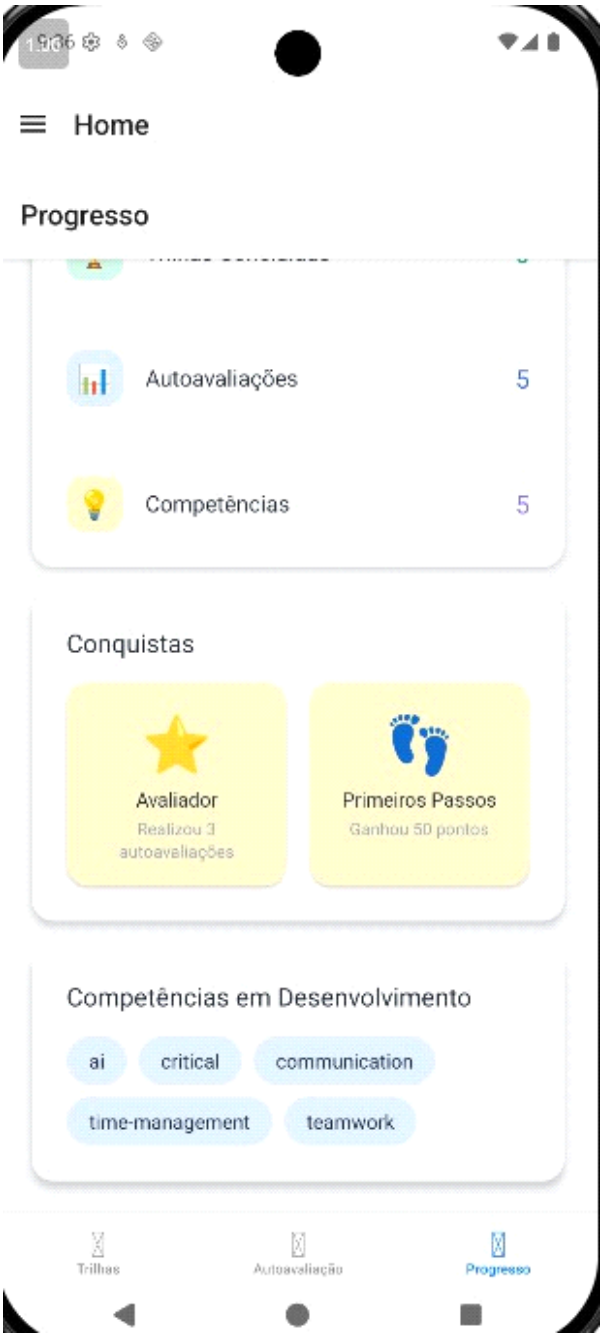

Home
Autoavaliação
Autoavaliação de Competências
Última avaliação desta competência: 9/10
Autoavaliação Enviada!
Competência: IA Básica
Nota: 9
+10 pontos ganhos!
Recomendações geradas com sucesso!
OK
Trilhas
Autoavaliação
Progresso

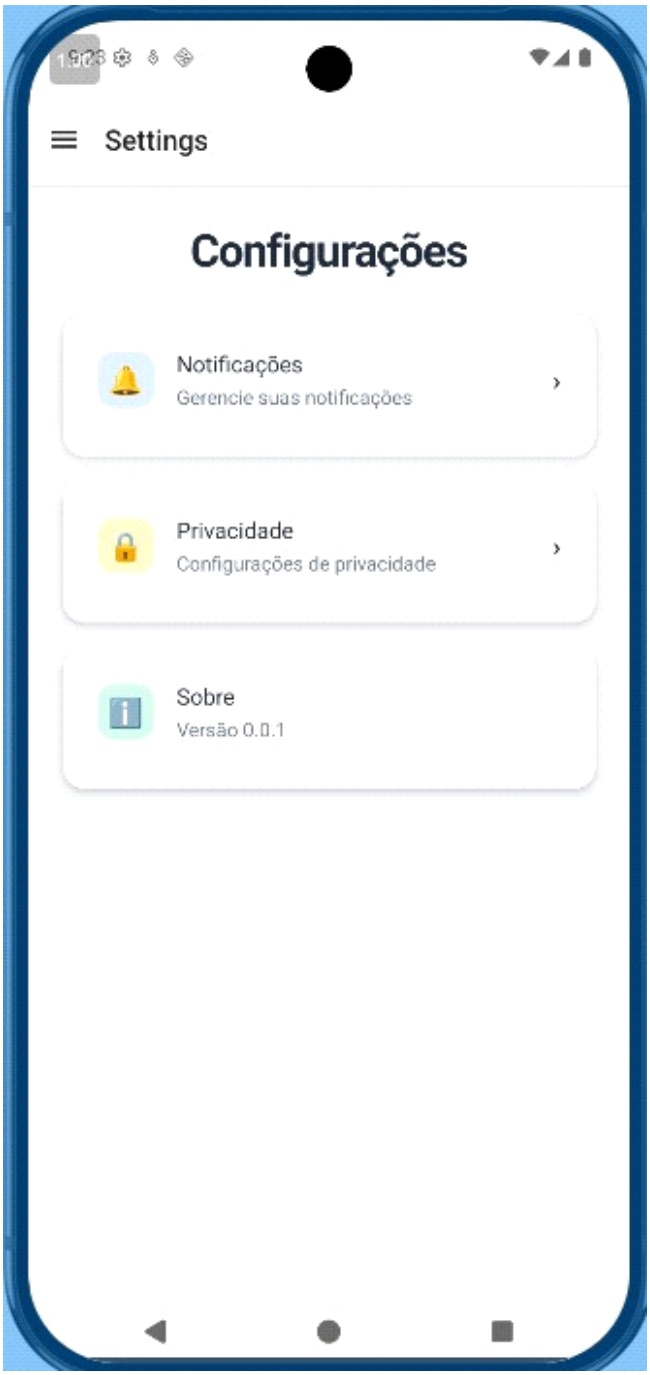

19:22
Profile
Meu Perfil
Informações Pessoais
Rafael
rafael@teste.com
SALVAR
Minhas Autoavaliações
Você ainda não realizou nenhuma autoavaliação.
Vá para a aba 'Autoavaliação' para começar!
SAIR


19:25
Profile
Meu Perfil
Informações Pessoais
Rafael
rafael@teste.com
SALVAR
Minhas Autoavaliações
IA Básica
19/11/2025
9/10
Resumo
Total de Avaliações:
1
Média Geral:
9.0/10
Competências Avaliadas:
1
SAIR


Home
Progresso
Meu Progresso
Total de Pontos
0
pontos
Estatísticas
Trilhas Concluídas
0
Autoavaliações
0
Competências
0
Conquistas
Trilhas
Autoavaliação
Progresso


19:34
Home
Progresso
Meu Progresso
Total de Pontos
10
pontos
Estatísticas
Trilhas Concluídas
0
Autoavaliações
1
Competências
1
Conquistas
Trilhas
Autoavaliação
Progresso


Home
Progresso
Autoavaliações
5
Competências
5
Conquistas
Avaliador
Realizou 3 autoavaliações
Primeiros Passos
Ganhou 50 pontos
Competências em Desenvolvimento
ai
critical
communication
time-management
teamwork
Trilhas
Autoavaliação
Progresso




**Códigos principais:**

## Códigos da Aplicação

**AssessmentScreen.jsx**

```
import React, { useState, useEffect } from 'react';
import {
  View,
  Text,
  ScrollView,
  TextInput,
  Button,
  Alert,
} from 'react-native';
import { Picker } from '@react-native-picker/picker';
import { stylesheet } from '../../../assets/stylesheet';
import { saveAssessment, getAssessmentBySkill, addPoints, checkBadges } from
'../../utils/storage';

type SkillOption = {
  label: string;
  value: string;
};

const SKILLS: SkillOption[] = [
  { label: 'Comunicação', value: 'communication' },
  { label: 'Pensamento Crítico', value: 'critical' },
  { label: 'IA Básica', value: 'ai' },
  { label: 'Sustentabilidade', value: 'sustain' },
  { label: 'Trabalho em Equipe', value: 'teamwork' },
  { label: 'Gestão do Tempo', value: 'time-management' },
];

export default function AssessmentScreen() {
  const [skill, setSkill] = useState<string>('communication');
  const [rating, setRating] = useState<string>('');
  const [lastRating, setLastRating] = useState<number | null>(null);

  useEffect(() => {
    loadLastAssessment();
  }, [skill]);

  const loadLastAssessment = async () => {
    const last = await getAssessmentBySkill(skill);
    if (last) {
      setLastRating(last.rating);
    } else {
      setLastRating(null);
    }
  };

  const submitAssessment = async () => {
    if (!rating || isNaN(Number(rating))) {
      Alert.alert('Erro', 'Digite uma nota de 0 a 10.');
      return;
    }

    const score = Number(rating);

    if (score < 0 || score > 10) {
      Alert.alert('Erro', 'A nota deve estar entre 0 e 10.');
      return;
    }
    const skillLabel = SKILLS.find((s) => s.value === skill)?.label || skill;

    try {
```

```
      await saveAssessment({
        skill,
        skillLabel,
        rating: score,
      });

      // Adicionar pontos por avaliação
      await addPoints(10);
      await checkBadges();

      Alert.alert(
        'Autoavaliação Enviada!',
        `Competência: ${skillLabel}\nNota: ${rating}\n\n+10 pontos ganhos!\n\nRecomendações
geradas com sucesso!`
      );

      setRating('');
      loadLastAssessment();
    } catch (error) {
      Alert.alert('Erro', 'Não foi possível salvar a avaliação.');
    }
  };

  return (
    <ScrollView contentContainerStyle={stylesheet.scrollContainer}>
      <Text style={stylesheet.title}>Autoavaliação de Competências</Text>

      {lastRating !== null && (
        <View style={[stylesheet.card, { backgroundColor: '#E3F2FD', marginBottom: 16 }]}>
          <Text style={[stylesheet.text, { fontWeight: '600' }]}>
            Última avaliação desta competência: {lastRating}/10
          </Text>
        </View>
      )}

      <Text style={stylesheet.label}>Selecione a Competência</Text>

      <View style={stylesheet.pickerWrapper}>
        <Picker selectedValue={skill} onValueChange={(value) => setSkill(value)}>
          {SKILLS.map((s) => (
            <Picker.Item key={s.value} label={s.label} value={s.value} />
          ))}
        </Picker>
      </View>

      <Text style={stylesheet.label}>Sua Nota (0-10)</Text>

      <TextInput
        style={stylesheet.input}
        keyboardType="numeric"
        placeholder="Ex: 7"
        value={rating}
        onChangeText={setRating}


      />

      <Button title="Enviar Avaliação" onPress={submitAssessment} />
    </ScrollView>
  );
}
```

## HomeTabs.tsx

```
import React from 'react';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { TabsParams } from '../../navigation/types';

import TrailsScreen from './TrailsScreen';
import AssessmentScreen from './AssessmentScreen';
import ProgressScreen from './ProgressScreen';
import TrailDetailsScreen from './TrailDetailsScreen';

const Tab = createBottomTabNavigator<TabsParams>();
const TrailsStack = createNativeStackNavigator<TabsParams>();

function TrailsStackNavigator() {
  return (
    <TrailsStack.Navigator>
      <TrailsStack.Screen
        name="Trails"
        component={TrailsScreen}
        options={{ title: 'Trilhas', headerShown: false }}
      />
      <TrailsStack.Screen
        name="TrailDetails"
        component={TrailDetailsScreen}
        options={{ title: 'Detalhes da Trilha' }}
      />
    </TrailsStack.Navigator>
  );
}

export default function HomeTabs() {
  return (
    <Tab.Navigator>
      <Tab.Screen name="Trails" component={TrailsStackNavigator} options={{ title: 'Trilhas',
headerShown: false }} />
      <Tab.Screen name="Assessment" component={AssessmentScreen} options={{ title:
'Autoavaliação' }} />
      <Tab.Screen name="Progress" component={ProgressScreen} options={{ title: 'Progresso' }}
/>
    </Tab.Navigator>
  );
}
```

## ProgressScreen.tsx

```
import React, { useState } from 'react';
import { View, Text, ScrollView } from 'react-native';
import { useFocusEffect } from '@react-navigation/native';
import { stylesheet, colors } from '../../../assets/stylesheet';
import { getUserProgress, getBadges } from '../../utils/storage';

const BADGE_INFO: Record<string, { name: string; emoji: string; description: string }> = {
  first_steps: {

    name: 'Primeiros Passos',
    emoji: '👣',
    description: 'Ganhou 50 pontos',
  },
```

```
  trail_complete: {
    name: 'Trilha Completa',
    emoji: '🏆',
    description: 'Completou sua primeira trilha',
  },
  assessor: {
    name: 'Avaliador',
    emoji: '⭐',
    description: 'Realizou 3 autoavaliações',
  },
  expert: {
    name: 'Especialista',
    emoji: '➵',
    description: 'Ganhou 200 pontos',
  },
};

export default function ProgressScreen() {
  const [progress, setProgress] = useState({
    totalPoints: 0,
    badges: [] as string[],
    completedTrails: 0,
    totalAssessments: 0,
    skillsDeveloped: [] as string[],
  });
  const [badges, setBadges] = useState<string[]>([]);

  useFocusEffect(
    React.useCallback(() => {
      loadProgress();
    }, [])
  );

  const loadProgress = async () => {
    const userProgress = await getUserProgress();
    const userBadges = await getBadges();
    setProgress(userProgress);
    setBadges(userBadges);
  };

  return (
    <ScrollView contentContainerStyle={stylesheet.scrollContainer}>
      <Text style={stylesheet.titleLarge}>Meu Progresso</Text>

      {/* Pontos */}
      <View style={[stylesheet.card, { backgroundColor: '#E3F2FD' }]}>
        <View style={{ flexDirection: 'row', alignItems: 'center', justifyContent: 'space-
between' }}>
          <View>
            <Text style={[stylesheet.text, { fontSize: 14 }]}>Total de Pontos</Text>
            <Text style={[stylesheet.title, { color: colors.primary, marginTop: 4 }]}>
              {progress.totalPoints} pts
            </Text>
          </View>
          <Text style={{ fontSize: 48 }}>⭐</Text>
        </View>

      </View>

      {/* Estatísticas */}
```

```
      <View style={stylesheet.card}>
        <Text style={[stylesheet.text, { fontWeight: '600', marginBottom: 12
}]}>Estatísticas</Text>


}}>







}}>

  <View style={{ flexDirection: 'row', justifyContent: 'space-between', marginBottom: 8

    <Text style={stylesheet.text}>Trilhas Concluídas</Text>
    <Text style={[stylesheet.text, { fontWeight: '700', color: colors.success }]}>
      {progress.completedTrails}
    </Text>
  </View>

  <View style={{ flexDirection: 'row', justifyContent: 'space-between', marginBottom: 8

    <Text style={stylesheet.text}>Autoavaliações Realizadas</Text>
    <Text style={[stylesheet.text, { fontWeight: '700', color: colors.primary }]}>
      {progress.totalAssessments}
    </Text>
  </View>

  <View style={{ flexDirection: 'row', justifyContent: 'space-between' }}>
    <Text style={stylesheet.text}>Competências Desenvolvidas</Text>
    <Text style={[stylesheet.text, { fontWeight: '700', color: colors.secondary }]}>
      {progress.skillsDeveloped.length}
    </Text>
  </View>
</View>


      {/* Badges */}
      <View style={stylesheet.card}>
        <Text style={[stylesheet.text, { fontWeight: '600', marginBottom: 12
}]}>Conquistas</Text>

        {badges.length === 0 ? (
          <Text style={stylesheet.metaText}>
            Complete trilhas e faça avaliações para ganhar conquistas!
          </Text>
        ) : (
          <View style={{ flexDirection: 'row', flexWrap: 'wrap', gap: 12 }}>
            {badges.map((badgeId) => {
              const badge = BADGE_INFO[badgeId];
              if (!badge) return null;
              return (
                <View
                  key={badgeId}
                  style={{
                    backgroundColor: '#FFF9C4',
                    padding: 12,
```

```
                    borderRadius: 8,
                    alignItems: 'center',
                    minWidth: 100,
                    marginBottom: 8,




'center' }]}>

  }}
>
  <Text style={{ fontSize: 32, marginBottom: 4 }}>{badge.emoji}</Text>
  <Text style={[stylesheet.text, { fontSize: 12, fontWeight: '600', textAlign:

    {badge.name}

                  </Text>
                  <Text style={[stylesheet.metaText, { fontSize: 10, textAlign: 'center' }]}>
                    {badge.description}
                  </Text>
                </View>
              );
            })}
          </View>
        )}
      </View>

      {/* Competências */}
      {progress.skillsDeveloped.length > 0 && (
        <View style={stylesheet.card}>
          <Text style={[stylesheet.text, { fontWeight: '600', marginBottom: 12 }]}>
            Competências em Desenvolvimento
          </Text>
          <View style={{ flexDirection: 'row', flexWrap: 'wrap', gap: 8 }}>
            {progress.skillsDeveloped.map((skill) => (
              <View
                key={skill}
                style={{
                  backgroundColor: colors.surface,
                  paddingHorizontal: 12,
                  paddingVertical: 6,
                  borderRadius: 16,
                  borderWidth: 1,
                  borderColor: colors.border,
                }}
              >
                <Text style={stylesheet.text}>{skill}</Text>
              </View>
            ))}
          </View>
        </View>
      )}
    </ScrollView>
  );
}
```

## TrailDetailsScreen.tsx

```
import React, { useState, useEffect } from 'react';
import {
  View,
  Text,
  ScrollView,
  TouchableOpacity,
  Alert,
} from 'react-native';
import { NativeStackScreenProps } from '@react-navigation/native-stack';
import { TabsParams } from '../../navigation/types';
import { getTrailById } from '../../utils/recommendations';
import { getTrailProgress, saveTrailProgress, completeTrail } from '../../utils/storage';
import { stylesheet, colors, spacing, radius, fonts, shadows } from
'../../../assets/stylesheet';

type Props = NativeStackScreenProps<TabsParams, 'TrailDetails'>;

export default function TrailDetailsScreen({ route, navigation }: Props) {

  const { trailId } = route.params;
  const [trail, setTrail] = useState(getTrailById(trailId));
  const [progress, setProgress] = useState(0);
  const [completed, setCompleted] = useState(false);

  useEffect(() => {
    loadProgress();
  }, [trailId]);

  const loadProgress = async () => {
    if (!trail) return;
    const trailProgress = await getTrailProgress(trailId);
    if (trailProgress) {
      setProgress(trailProgress.progress);
      setCompleted(trailProgress.completed);
    }
  };

  const startTrail = async () => {
    if (!trail) return;
    await saveTrailProgress({
      trailId,
      completed: false,
      progress: 0,
      startedAt: new Date().toISOString(),
    });
    setProgress(0);
    Alert.alert('Trilha Iniciada!', 'Boa sorte na sua jornada de aprendizado!');
  };

  const completeLesson = async (lessonId: string) => {
    if (!trail) return;
    const lessonIndex = trail.lessons.findIndex((l) => l.id === lessonId);
    if (lessonIndex === -1) return;

    const newProgress = Math.min(100, ((lessonIndex + 1) / trail.lessons.length) * 100);
    await saveTrailProgress({
      trailId,
      completed: newProgress === 100,
      progress: newProgress,
      startedAt: (await getTrailProgress(trailId))?.startedAt || new Date().toISOString(),
    });
```

```
    setProgress(newProgress);

    if (newProgress === 100) {

      await completeTrail(trailId);
      setCompleted(true);
      Alert.alert(
        'Parabéns! 🚀',
        'Você completou esta trilha!\n+50 pontos ganhos!'
      );
    }
  };

  if (!trail) {
    return (
      <View style={stylesheet.centeredContainer}>
        <Text style={stylesheet.text}>Trilha não encontrada</Text>
      </View>
    );

  }
  const completedLessons = trail.lessons.filter((l) => l.completed).length;

  return (

    <ScrollView contentContainerStyle={stylesheet.scrollContainer}>
      <Text style={stylesheet.titleLarge}>{trail.title}</Text>

      <View style={stylesheet.card}>
        <Text style={stylesheet.text}>{trail.description}</Text>
        <View style={stylesheet.trailMetaRow}>
          <Text style={[stylesheet.metaText, stylesheet.trailMetaItem]}>
            ⏱ {trail.duration}
          </Text>
          <Text style={[stylesheet.metaText, stylesheet.trailMetaItem]}>
            📶 {trail.level}
          </Text>
          <Text style={stylesheet.metaText}>
            💼 {trail.category}
          </Text>
        </View>
      </View>

      {progress > 0 && (
        <View style={stylesheet.card}>
          <View style={stylesheet.progressRow}>
            <Text style={stylesheet.textBold}>Progresso</Text>
            <Text style={[stylesheet.textBold, { color: colors.primary
}]}>{Math.round(progress)}%</Text>
          </View>
          <View style={stylesheet.progressBar}>
            <View
              style={[
                stylesheet.progressFill,
                { width: `${progress}%` },
              ]}
            />
          </View>
          <Text style={[stylesheet.metaText, { marginTop: spacing.sm }]}>
            {completedLessons} de {trail.lessons.length} lições concluídas
          </Text>
```

```
        </View>
      )}

      <Text style={[stylesheet.title, { marginTop: 8 }]}>Lições</Text>

      {trail.lessons.map((lesson, index) => {
        const isCompleted = lesson.completed || (progress > 0 && index < completedLessons);
        const isLocked = progress === 0 && index > 0;

        return (
          <TouchableOpacity
            key={lesson.id}
            style={[
              stylesheet.card,
              isCompleted && stylesheet.lessonCompleted,
              isLocked && stylesheet.lessonLocked,
            ]}
            onPress={() => {
              if (isLocked) {
                Alert.alert('Lição Bloqueada', 'Complete as lições anteriores primeiro.');

                return;
              }
              if (isCompleted) {
                Alert.alert('Lição Concluída', 'Você já completou esta lição!');
                return;
              }
              Alert.alert(
                'Lição',
                `Você completou: ${lesson.title}\n\nTipo: ${lesson.type === 'video' ? 'Vídeo'
: lesson.type === 'text' ? 'Texto' : 'Quiz'}\nDuração: ${lesson.duration}`,
                [
                  { text: 'Cancelar', style: 'cancel' },
                  {
                    text: 'Marcar como Concluída',
                    onPress: () => completeLesson(lesson.id),
                  },
                ]
              );
            }}
            disabled={isLocked}
            activeOpacity={0.7}
          >
            <View style={stylesheet.lessonRow}>
              <View style={[
                stylesheet.lessonIconContainer,
                isCompleted && stylesheet.lessonIconCompleted,
                isLocked && stylesheet.lessonIconLocked,
                !isCompleted && !isLocked && stylesheet.lessonIconDefault,
              ]}>
                <Text style={stylesheet.lessonEmoji}>
                  {isCompleted ? '✓' : isLocked ? 'A' : '🔓'}
                </Text>
              </View>
              <View style={stylesheet.lessonContent}>
                <Text style={[stylesheet.textBold, isCompleted && { color: colors.success }]}>
                  {lesson.title}
                </Text>
                <Text style={[stylesheet.metaText, { marginTop: spacing.xs }]}>
                  {lesson.type === 'video' ? '📹 Vídeo' : lesson.type === 'text' ? '📄 Texto'
```

```
: '? Quiz'} • {lesson.duration}
                </Text>
              </View>
            </View>
          </TouchableOpacity>
        );
      })}

      {progress === 0 && (
        <TouchableOpacity
          style={[stylesheet.button, { marginTop: spacing.lg }]}
          onPress={startTrail}
          activeOpacity={0.8}
        >
          <Text style={stylesheet.buttonText}>Iniciar Trilha</Text>
        </TouchableOpacity>
      )}

      {completed && (
        <View style={[stylesheet.card, stylesheet.completionCard]}>
          <Text style={[stylesheet.textBold, stylesheet.completionText]}>

            🎉 Parabéns! Você completou esta trilha!
          </Text>
        </View>
      )}
    </ScrollView>
  );
}
```

## TrailsScreen.tsx

```
import React, { useState } from 'react';
import { ScrollView, Text, View, TouchableOpacity } from 'react-native';
import { useFocusEffect } from '@react-navigation/native';
import TrailCard from '../../components/TrailCard';
import { stylesheet } from '../../../assets/stylesheet';
import { getRecommendedTrails, getAllTrails, Trail } from '../../utils/recommendations';
import { NativeStackScreenProps } from '@react-navigation/native-stack';
import { TabsParams } from '../../navigation/types';
type Props = NativeStackScreenProps<TabsParams, 'Trails'>;

export default function TrailsScreen({ navigation }: Props) {

  const [recommendedTrails, setRecommendedTrails] = useState<Trail[]>([]);
  const [allTrails, setAllTrails] = useState<Trail[]>([]);
  const [showAll, setShowAll] = useState(false);

  useFocusEffect(
    React.useCallback(() => {
      loadTrails();
    }, [])
  );

  const loadTrails = async () => {
    const recommended = await getRecommendedTrails();
    const all = getAllTrails();
    setRecommendedTrails(recommended);
    setAllTrails(all);
```

```
  };
  const trailsToShow = showAll ? allTrails : recommendedTrails;

  return (
    <ScrollView contentContainerStyle={stylesheet.scrollContainer}>
      <View style={{ flexDirection: 'row', justifyContent: 'space-between', alignItems:
'center', marginBottom: 12 }}>
        <Text style={stylesheet.title}>
          {showAll ? 'Todas as Trilhas' : 'Trilhas Recomendadas'}
        </Text>
        <TouchableOpacity
          onPress={() => setShowAll(!showAll)}
          style={{ padding: 8 }}
        >
          <Text style={stylesheet.linkText}>
            {showAll ? 'Ver Recomendadas' : 'Ver Todas'}
          </Text>
        </TouchableOpacity>
      </View>

      {!showAll && recommendedTrails.length > 0 && (
        <View style={[stylesheet.card, { backgroundColor: '#E8F5E9', marginBottom: 16 }]}>

          <Text style={[stylesheet.text, { fontWeight: '600' }]}>
            • Baseado nas suas autoavaliações, recomendamos estas trilhas para você!
          </Text>
        </View>
      )}

      {trailsToShow.length === 0 ? (
        <View style={stylesheet.card}>
          <Text style={stylesheet.text}>Nenhuma trilha disponível no momento.</Text>
        </View>
      ) : (
        trailsToShow.map((trail) => (
          <TrailCard
            key={trail.id}
            trail={trail}
            onPress={() => navigation.navigate('TrailDetails', { trailId: trail.id })}
          />
        ))
      )}
    </ScrollView>
  );
}
```

## LoginScreen.tsx

```
import React, { useState } from 'react';
import { View, Text, TextInput, Button, TouchableOpacity, Alert } from 'react-native';
import { NativeStackScreenProps } from '@react-navigation/native-stack';
import { AuthStackParams } from '../../navigation/types';
import { saveToken, saveProfile } from '../../utils/storage';
import { stylesheet } from '../../../assets/stylesheet';
type Props = NativeStackScreenProps<AuthStackParams, 'Login'>;

export default function LoginScreen({ navigation }: Props) {

  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
```

```
  const handleLogin = async () => {
    if (!email || !password) {
      Alert.alert('Erro', 'Informe email e senha');
      return;
    }

    const token = 'token_' + Date.now();
    await saveToken(token);
    await saveProfile({ name: 'Usuário', email });

    // Navigate to App drawer after successful login
    navigation.getParent()?.reset({
      index: 0,
      routes: [{ name: 'App' }],
    });
  };

  return (
    <View style={stylesheet.container}>
      <Text style={stylesheet.titleLarge}>SkillUpPlus 2030+</Text>

      <TextInput
        style={stylesheet.input}

        placeholder="Email"
        autoCapitalize="none"
        value={email}
        onChangeText={setEmail}
      />
      <TextInput
        style={stylesheet.input}
        placeholder="Senha"
        secureTextEntry
        value={password}
        onChangeText={setPassword}
      />

      <Button title="Entrar" onPress={handleLogin} />

      <TouchableOpacity onPress={() => navigation.navigate('Register')}
style={stylesheet.link}>
        <Text style={stylesheet.linkText}>Criar conta</Text>
      </TouchableOpacity>
    </View>
  );
}
```

## RegisterScreen.tsx

```
import React, { useState } from 'react';
import { View, Text, TextInput, Button, Alert } from 'react-native';
import { NativeStackScreenProps } from '@react-navigation/native-stack';
import { AuthStackParams } from '../../navigation/types';
import { saveToken, saveProfile } from '../../utils/storage';
import { stylesheet } from '../../../assets/stylesheet';
type Props = NativeStackScreenProps<AuthStackParams, 'Register'>;

export default function RegisterScreen({ navigation }: Props) {

  const [name, setName] = useState('');
```

```
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleRegister = async () => {
    if (!name || !email || !password) {
      Alert.alert('Erro', 'Preencha todos os campos');
      return;
    }

    const token = 'token_' + Date.now();
    await saveToken(token);
    await saveProfile({ name, email });

    // Navigate to App drawer after successful registration
    navigation.getParent()?.reset({
      index: 0,
      routes: [{ name: 'App' }],
    });
  };

  return (
    <View style={stylesheet.container}>
      <Text style={stylesheet.titleLarge}>Criar conta</Text>

      <TextInput

        style={stylesheet.input}
        placeholder="Nome"
        value={name}
        onChangeText={setName}
      />
      <TextInput
        style={stylesheet.input}
        placeholder="Email"
        value={email}
        onChangeText={setEmail}
        autoCapitalize="none"
      />
      <TextInput
        style={stylesheet.input}
        placeholder="Senha"
        secureTextEntry
        value={password}
        onChangeText={setPassword}
      />

      <Button title="Registrar" onPress={handleRegister} />
    </View>
  );
}
```

## mainNavigator.tsx

```
import React from 'react';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { createDrawerNavigator } from '@react-navigation/drawer';
import { RootStackParams, AuthStackParams, AppDrawerParams } from './types';

import LoginScreen from '../screens/Auth/LoginScreen';

import RegisterScreen from '../screens/Auth/RegisterScreen';
```

```
import HomeTabs from '../screens/Home/HomeTabs';
import ProfileScreen from '../screens/ProfileScreen';
import SettingsScreen from '../screens/SettingsScreen';

const RootStack = createNativeStackNavigator<RootStackParams>();
const AuthStack = createNativeStackNavigator<AuthStackParams>();
const Drawer = createDrawerNavigator<AppDrawerParams>();

function AuthRoutes() {
  return (
    <AuthStack.Navigator>
      <AuthStack.Screen name="Login" component={LoginScreen} options={{ title: 'Login' }} />
      <AuthStack.Screen name="Register" component={RegisterScreen} options={{ title:
'Registrar' }} />
    </AuthStack.Navigator>
  );
}

function AppDrawer() {
  return (
    <Drawer.Navigator>
      <Drawer.Screen name="Home" component={HomeTabs} />
      <Drawer.Screen name="Profile" component={ProfileScreen} />
      <Drawer.Screen name="Settings" component={SettingsScreen} />
    </Drawer.Navigator>

  );
}

export default function MainNavigator({ initialRouteName }: { initialRouteName: 'Auth' | 'App'
}) {
  return (
    <RootStack.Navigator screenOptions={{ headerShown: false }}
initialRouteName={initialRouteName}>
      <RootStack.Screen name="Auth" component={AuthRoutes} />
      <RootStack.Screen name="App" component={AppDrawer} />
    </RootStack.Navigator>
  );
}
```

## types.ts

```
export type AuthStackParams = {
  Login: undefined;
  Register: undefined;
};

export type AppDrawerParams = {
  Home: undefined;
  Profile: undefined;
  Settings: undefined;
};

export type TabsParams = {
  Trails: undefined;
  Assessment: undefined;
  Progress: undefined;
  TrailDetails: { trailId: string };
};
```

```
export type RootStackParams = {
  Auth: undefined;
  App: undefined;
};
```

## TrailCard.tsx

```
import React from 'react';
import { View, Text, TouchableOpacity, Image } from 'react-native';
import { stylesheet } from '../../assets/stylesheet';
import { Trail } from '../utils/recommendations';

interface Props {
  trail: Trail;
  onPress?: () => void;
}

export default function TrailCard({ trail, onPress }: Props) {
  return (
    <TouchableOpacity
      style={stylesheet.trailCard}
      onPress={onPress}
    >
      {trail.image && (
        <Image
          source={trail.image}
          style={stylesheet.trailCardImage}
        />

      )}
      <View style={stylesheet.trailCardContent}>
        <Text style={stylesheet.text}>{trail.title}</Text>
        <Text style={stylesheet.metaText}>{trail.level} • {trail.duration} •
{trail.category}</Text>
        {trail.description && (
          <Text style={[stylesheet.metaText, { marginTop: 4 }]} numberOfLines={2}>
            {trail.description}
          </Text>
        )}
      </View>
    </TouchableOpacity>
  );
}
```