Layout

// app/\_layout.tsx

import { Stack } from 'expo-router'

import { TamaguiProvider, Theme } from 'tamagui'

import tamagui from '../tamagui.config'

export default function Layout() {

  return (

    <TamaguiProvider config={tamagui}>

      {/\* thème global (change en "dark" si tu veux forcer le sombre) \*/}

      <Theme name="light">

        <Stack screenOptions={{ headerShown: false }}>

          {/\* Accueil (hero + bouton ENTRER) \*/}

          <Stack.Screen name="index" />

          {/\* Sélection des ingrédients \*/}

          <Stack.Screen

            name="ingredients"

            options={{ headerShown: true, title: 'Ingrédients' }}

          />

          {/\* Écran convertisseur (kawaii) \*/}

          <Stack.Screen

            name="convertisseur"

            options={{ headerShown: true, title: 'Convertisseur' }}

          />

        </Stack>

      </Theme>

    </TamaguiProvider>

  )

}

Convertisseur

import { router } from 'expo-router'

import React from 'react'

import { StyleSheet, Text, TextInput, TouchableOpacity, View } from 'react-native'

export default function ConvertisseurScreen() {

  return (

    <View style={styles.container}>

      <Text style={styles.title}>✨ Convertisseur Kawaii 💖</Text>

      <View style={styles.card}>

        <Text style={styles.label}>Valeur à convertir</Text>

        <TextInput style={styles.input} placeholder="ex: 2 c. à s." placeholderTextColor="#ff8fcd" />

        <Text style={styles.label}>Ingrédient</Text>

        <TextInput style={styles.input} placeholder="ex: farine" placeholderTextColor="#ff8fcd" />

        <TouchableOpacity style={styles.cta} onPress={() => {}}>

          <Text style={styles.ctaText}>💫 Convertir</Text>

        </TouchableOpacity>

      </View>

      <TouchableOpacity onPress={() => router.back()} style={{ marginTop: 16 }}>

        <Text style={{ color: '#ff4fa2', fontWeight: '700' }}>⬅️ Retour</Text>

      </TouchableOpacity>

    </View>

  )

}

const styles = StyleSheet.create({

  container: { flex: 1, backgroundColor: '#FFEEFC', alignItems: 'center', justifyContent: 'center', padding: 20 },

  title: { fontSize: 26, fontWeight: '900', color: '#FF4FA2', marginBottom: 20, textShadowColor: '#fff', textShadowOffset: { width: 1, height: 1 }, textShadowRadius: 3 },

  card: { width: '92%', backgroundColor: '#fff', borderRadius: 24, padding: 20, shadowColor: '#FF8FCD', shadowOpacity: 0.2, shadowRadius: 10, elevation: 8 },

  label: { fontSize: 15, fontWeight: '700', color: '#FF69B4', marginBottom: 6 },

  input: { backgroundColor: '#FFF0FA', borderRadius: 14, borderWidth: 2, borderColor: '#FFB6F9', padding: 12, fontSize: 16, color: '#FF4FA2', marginBottom: 14 },

  cta: { backgroundColor: '#FF92E0', borderRadius: 999, paddingVertical: 14, alignItems: 'center', elevation: 4, shadowColor: '#FF4FA2', shadowOpacity: 0.3, shadowRadius: 6 },

  ctaText: { color: '#fff', fontWeight: '900', fontSize: 18 },

})

Index

// app/index.tsx

import { router } from 'expo-router'

import React from 'react'

import { Image, StyleSheet, Text, TouchableOpacity, useWindowDimensions, View } from 'react-native'

export default function HomeScreen() {

  const { height, width } = useWindowDimensions()

  // image = ~60% de la hauteur écran, bornée

  const IMG\_H = Math.round(Math.min(540, Math.max(320, height \* 0.60)))

  const go = () => router.push('/ingredients') // nouvelle destination

  return (

    <View style={styles.container}>

      {/\* IMAGE – hauteur explicite pour éviter toute disparition \*/}

      <TouchableOpacity style={[styles.imageWrapper, { height: IMG\_H }]} onPress={go} activeOpacity={0.9}>

        <Image

          // ⚠️ adapte l’extension si nécessaire (.webp/.png/.jpg)

          source={require('../assets/hero-splash.webp')}

          style={styles.image}

          resizeMode="contain"   // ne coupe ni haut ni bas

        />

      </TouchableOpacity>

      {/\* BOUTON – plus haut (moins d’espace bas) \*/}

      <TouchableOpacity style={styles.button} onPress={go} activeOpacity={0.9}>

        <Text style={styles.buttonText}>ENTRER</Text>

      </TouchableOpacity>

    </View>

  )

}

const ORANGE\_FOND = '#F7A14D'

const BEIGE\_BTN   = '#FBE9D0'

const BRUN        = '#6B3E1D'

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: ORANGE\_FOND,

    alignItems: 'center',

    justifyContent: 'center',

    paddingHorizontal: 20,

    paddingTop: 24,

    paddingBottom: 12, // 👈 moins d’espace bas

  },

  imageWrapper: {

    width: '100%',

    marginTop: 8,

    marginBottom: 12,  // 👈 bouton remonte

    borderRadius: 16,

    overflow: 'hidden',

  },

  image: {

    width: '100%',

    height: '100%',

  },

  button: {

    backgroundColor: BEIGE\_BTN,

    borderRadius: 28,

    borderWidth: 2,

    borderColor: BRUN,

    paddingVertical: 10,     // fin

    paddingHorizontal: 56,   // allongé

    // remonte visuellement le bouton

    marginBottom: 8,

    // petite ombre douce

    shadowColor: '#000',

    shadowOpacity: 0.12,

    shadowRadius: 10,

    elevation: 6,

  },

  buttonText: {

    fontSize: 18,

    fontWeight: '800',

    color: BRUN,

    letterSpacing: 0.5,

  },

})

Ingredients

// app/ingredients.tsx

import { router } from 'expo-router'

import React, { useMemo, useState } from 'react'

import { ScrollView, StyleSheet, Text, TextInput, TouchableOpacity, View } from 'react-native'

const DB = [

  'Farine','Sucre','Sel','Beurre','Huile','Lait',

  'Riz','Pâtes','Cacao','Amande','Noisette',

  'Carotte','Pomme','Poire','Oignon','Ail','Tomate',

  'Pommes de terre','Courgette','Poivron','Citron'

]

export default function IngredientsScreen() {

  const [q, setQ] = useState('')

  const [sel, setSel] = useState<string[]>([])

  const list = useMemo(() => {

    const s = q.trim().toLowerCase()

    return s ? DB.filter(x => x.toLowerCase().includes(s)) : DB

  }, [q])

  const toggle = (name: string) =>

    setSel(prev => prev.includes(name) ? prev.filter(x => x !== name) : [...prev, name])

  const validate = () => {

    // plus tard: router.push({ pathname:'/convertisseur', params:{ items: JSON.stringify(sel) }})

    router.push('/convertisseur') // pour l’instant on enchaîne vers l’écran suivant

  }

  return (

    <View style={st.container}>

      <Text style={st.title}>🧺 Ingrédients</Text>

      <View style={st.card}>

        <TextInput

          style={st.input}

          value={q}

          onChangeText={setQ}

          placeholder="Rechercher un ingrédient…"

          placeholderTextColor="#ff8fcd"

        />

        <ScrollView contentContainerStyle={st.chipsWrap} keyboardShouldPersistTaps="handled">

          {list.map((name) => {

            const on = sel.includes(name)

            return (

              <TouchableOpacity

                key={name}

                onPress={() => toggle(name)}

                activeOpacity={0.9}

                style={[st.chip, on && st.chipOn]}

              >

                <Text style={[st.chipText, on && st.chipTextOn]}>{name}</Text>

              </TouchableOpacity>

            )

          })}

        </ScrollView>

        <TouchableOpacity

          onPress={validate}

          disabled={sel.length === 0}

          activeOpacity={0.9}

          style={[st.cta, sel.length === 0 && { opacity: 0.5 }]}

        >

          <Text style={st.ctaText}>Valider ({sel.length})</Text>

        </TouchableOpacity>

      </View>

    </View>

  )

}

const st = StyleSheet.create({

  container: { flex: 1, backgroundColor: '#FFEEFC', padding: 18, paddingTop: 36 },

  title: { fontSize: 26, fontWeight: '900', color: '#FF4FA2', marginBottom: 14 },

  card: {

    flex: 1, backgroundColor: '#fff', borderRadius: 22, padding: 14,

    shadowColor: '#FF8FCD', shadowOpacity: 0.18, shadowRadius: 10, elevation: 6

  },

  input: {

    backgroundColor: '#FFF0FA', borderRadius: 14, borderWidth: 2, borderColor: '#FFB6F9',

    paddingHorizontal: 14, paddingVertical: 10, fontSize: 16, color: '#FF4FA2', marginBottom: 10

  },

  chipsWrap: { flexDirection: 'row', flexWrap: 'wrap', gap: 10, paddingVertical: 6 },

  chip: {

    paddingVertical: 10, paddingHorizontal: 14,

    backgroundColor: '#FFE4F6', borderRadius: 999, borderWidth: 2, borderColor: '#FFB6F9'

  },

  chipOn: { backgroundColor: '#FF92E0', borderColor: '#FF4FA2' },

  chipText: { color: '#FF4FA2', fontWeight: '700' },

  chipTextOn: { color: '#fff' },

  cta: {

    marginTop: 14, backgroundColor: '#FF92E0', borderRadius: 999, alignItems: 'center',

    paddingVertical: 14, shadowColor: '#FF4FA2', shadowOpacity: 0.28, shadowRadius: 6, elevation: 5

  },

  ctaText: { color: '#fff', fontSize: 18, fontWeight: '900' },

})

Results

import { router, useLocalSearchParams } from 'expo-router'

import React, { useMemo } from 'react'

import { Button, Card, Separator, Text, XStack, YStack } from 'tamagui'

// Même base que dans select (tu peux later factoriser dans /constants)

const DB: Record<string, {

  label: string

  gPerTbsp: number

  gPerTsp: number

  avgUnitG: number

  peeledFactor: number

}> = {

  farine:  { label:'Farine',       gPerTbsp:8,  gPerTsp:3,  avgUnitG:0,   peeledFactor:1.00 },

  sucre:   { label:'Sucre',        gPerTbsp:12, gPerTsp:4,  avgUnitG:0,   peeledFactor:1.00 },

  sel:     { label:'Sel fin',      gPerTbsp:18, gPerTsp:6,  avgUnitG:0,   peeledFactor:1.00 },

  huile:   { label:'Huile',        gPerTbsp:14, gPerTsp:5,  avgUnitG:0,   peeledFactor:1.00 },

  lait:    { label:'Lait',         gPerTbsp:15, gPerTsp:5,  avgUnitG:0,   peeledFactor:1.00 },

  riz:     { label:'Riz cru',      gPerTbsp:12, gPerTsp:4,  avgUnitG:0,   peeledFactor:1.00 },

  beurre:  { label:'Beurre',       gPerTbsp:14, gPerTsp:5,  avgUnitG:0,   peeledFactor:1.00 },

  carotte: { label:'Carotte',      gPerTbsp:0,  gPerTsp:0,  avgUnitG:60,  peeledFactor:0.90 },

  pomme:   { label:'Pomme',        gPerTbsp:0,  gPerTsp:0,  avgUnitG:180, peeledFactor:0.90 },

  oignon:  { label:'Oignon',       gPerTbsp:0,  gPerTsp:0,  avgUnitG:110, peeledFactor:0.88 },

  ail:     { label:'Gousse d’ail', gPerTbsp:0,  gPerTsp:0,  avgUnitG:4,   peeledFactor:0.95 },

}

function Row({ left, right }: { left: string; right: string }) {

  return (

    <XStack ai="center" jc="space-between">

      <Text>{left}</Text>

      <Text fontWeight="700">{right}</Text>

    </XStack>

  )

}

export default function Results() {

  const { items } = useLocalSearchParams<{ items?: string }>()

  const ids: string[] = useMemo(() => {

    try { return items ? JSON.parse(items) : [] } catch { return [] }

  }, [items])

  return (

    <YStack f={1} bg="$background" p="$4" gap="$4">

      <XStack ai="center" jc="space-between">

        <Text fontSize={22} fontWeight="800">Conversions</Text>

        <Button size="$3" variant="outlined" onPress={() => router.back()}>↩︎ Modifier</Button>

      </XStack>

      {ids.length === 0 && (

        <Text>Pas d’ingrédients sélectionnés.</Text>

      )}

      <YStack gap="$4" pb="$6">

        {ids.map((id) => {

          const d = DB[id]

          if (!d) return null

          const tbsp = d.gPerTbsp ? `${d.gPerTbsp} g` : '—'

          const tsp  = d.gPerTsp  ? `${d.gPerTsp} g`  : '—'

          const unit = d.avgUnitG ? `${d.avgUnitG} g` : '—'

          const peeled =

            d.avgUnitG && d.peeledFactor

              ? `${Math.round(d.avgUnitG \* d.peeledFactor)} g`

              : '—'

          return (

            <Card key={id} p="$4" br="$8" bordered elevate>

              <Text fontWeight="800" fontSize={18} mb="$3">{d.label}</Text>

              <YStack gap="$2">

                <Row left="1 c. à soupe ↔ g" right={tbsp} />

                <Row left="1 c. à café ↔ g"  right={tsp} />

                <Row left="1 pièce moyenne"  right={unit} />

                <Row left="Épluché (par pièce)" right={peeled} />

              </YStack>

              <Separator my="$3" />

              {/\* Exemples rapides utiles \*/}

              <YStack gap="$1">

                {!!d.gPerTbsp && <Text>2 c. à soupe ≈ <Text fontWeight="700">{d.gPerTbsp \* 2} g</Text></Text>}

                {!!d.gPerTsp  && <Text>3 c. à café ≈ <Text fontWeight="700">{d.gPerTsp \* 3} g</Text></Text>}

              </YStack>

            </Card>

          )

        })}

      </YStack>

    </YStack>

  )

}

Select

import { router } from 'expo-router'

import React, { useMemo, useState } from 'react'

import { Button, Card, Input, ScrollView, Text, XStack, YStack } from 'tamagui'

// Mini base d’ingrédients (exemples)

const INGREDIENTS = [

  { id: 'farine',      label: 'Farine',        gPerTbsp: 8,  gPerTsp: 3,  avgUnitG: 0,   peeledFactor: 1.00 },

  { id: 'sucre',       label: 'Sucre',         gPerTbsp: 12, gPerTsp: 4,  avgUnitG: 0,   peeledFactor: 1.00 },

  { id: 'sel',         label: 'Sel fin',       gPerTbsp: 18, gPerTsp: 6,  avgUnitG: 0,   peeledFactor: 1.00 },

  { id: 'huile',       label: 'Huile',         gPerTbsp: 14, gPerTsp: 5,  avgUnitG: 0,   peeledFactor: 1.00 },

  { id: 'lait',        label: 'Lait',          gPerTbsp: 15, gPerTsp: 5,  avgUnitG: 0,   peeledFactor: 1.00 },

  { id: 'riz',         label: 'Riz cru',       gPerTbsp: 12, gPerTsp: 4,  avgUnitG: 0,   peeledFactor: 1.00 },

  { id: 'beurre',      label: 'Beurre',        gPerTbsp: 14, gPerTsp: 5,  avgUnitG: 0,   peeledFactor: 1.00 },

  { id: 'carotte',     label: 'Carotte',       gPerTbsp: 0,  gPerTsp: 0,  avgUnitG: 60,  peeledFactor: 0.90 },

  { id: 'pomme',       label: 'Pomme',         gPerTbsp: 0,  gPerTsp: 0,  avgUnitG: 180, peeledFactor: 0.90 },

  { id: 'oignon',      label: 'Oignon',        gPerTbsp: 0,  gPerTsp: 0,  avgUnitG: 110, peeledFactor: 0.88 },

  { id: 'ail',         label: 'Gousse d’ail',  gPerTbsp: 0,  gPerTsp: 0,  avgUnitG: 4,   peeledFactor: 0.95 },

]

export default function Select() {

  const [q, setQ] = useState('')

  const [sel, setSel] = useState<string[]>([])

  const list = useMemo(() => {

    const s = q.trim().toLowerCase()

    return !s ? INGREDIENTS

      : INGREDIENTS.filter(i => i.label.toLowerCase().includes(s) || i.id.includes(s))

  }, [q])

  const toggle = (id: string) =>

    setSel(prev => prev.includes(id) ? prev.filter(x => x !== id) : [...prev, id])

  const goResults = () => {

    const payload = JSON.stringify(sel)

    router.push({ pathname: '/results', params: { items: payload } })

  }

  return (

    <YStack f={1} bg="$background" p="$4" gap="$4">

      <Text fontSize={22} fontWeight="800">Sélection d’ingrédients</Text>

      {/\* Recherche \*/}

      <Input

        placeholder="Rechercher… (ex: farine, sucre, carotte)"

        value={q}

        onChangeText={setQ}

      />

      {/\* Sélection en cours \*/}

      {sel.length > 0 && (

        <XStack flexWrap="wrap" gap="$2">

          {sel.map(id => {

            const it = INGREDIENTS.find(i => i.id === id)!

            return (

              <Button

                key={id}

                size="$3"

                bg="$blue10"

                px="$3"

                br="$8"

                onPress={() => toggle(id)}

              >

                <Text color="white">{it.label} ✕</Text>

              </Button>

            )

          })}

        </XStack>

      )}

      {/\* Liste filtrée \*/}

      <ScrollView style={{ flex: 1 }}>

        <YStack gap="$3" pb="$6">

          {list.map(it => {

            const on = sel.includes(it.id)

            return (

              <Card

                key={it.id}

                p="$3"

                br="$8"

                bordered

                bg={on ? '$blue4' : '$background'}

                pressStyle={{ scale: 0.98 }}

                onPress={() => toggle(it.id)}

              >

                <XStack ai="center" jc="space-between">

                  <Text fontSize={16} fontWeight="700">{it.label}</Text>

                  <Button size="$3" variant={on ? 'solid' : 'outlined'}>

                    {on ? 'Ajouté' : 'Ajouter'}

                  </Button>

                </XStack>

              </Card>

            )

          })}

        </YStack>

      </ScrollView>

      {/\* Valider \*/}

      <Button

        bg="$blue10"

        br="$10"

        px="$6"

        py="$3"

        disabled={sel.length === 0}

        opacity={sel.length === 0 ? 0.5 : 1}

        onPress={goResults}

      >

        <Text color="white" fontWeight="800">Valider ({sel.length})</Text>

      </Button>

    </YStack>

  )

}

App.json

{

  "expo": {

    "name": "Convertisseur",

    "slug": "Convertisseur",

    "version": "1.0.0",

    "orientation": "portrait",

    "icon": "./assets/images/icon.png",

    "scheme": "convertisseur",

    "userInterfaceStyle": "automatic",

    "newArchEnabled": true,

    "ios": {

      "supportsTablet": true

    },

    "android": {

      "adaptiveIcon": {

        "foregroundImage": "./assets/images/adaptive-icon.png",

        "backgroundColor": "#ffffff"

      },

      "edgeToEdgeEnabled": true

    },

    "web": {

      "bundler": "metro",

      "output": "static",

      "favicon": "./assets/images/favicon.png"

    },

    "plugins": [

      "expo-router",

      [

        "expo-splash-screen",

        {

          "image": "./assets/images/splash-icon.png",

          "imageWidth": 200,

          "resizeMode": "contain",

          "backgroundColor": "#ffffff"

        }

      ]

    ],

    "experiments": {

      "typedRoutes": true

    }

  }

}

Babel.config.json

module.exports = function (api) {

  api.cache(true)

  return {

    presets: ['babel-preset-expo'],

    plugins: [

      ['@tamagui/babel-plugin', { config: './tamagui.config.ts' }],

      'react-native-reanimated/plugin' // toujours en dernier

    ],

  }

}

Package.json

{

  "name": "convertisseur",

  "main": "expo-router/entry",

  "version": "1.0.0",

  "scripts": {

    "start": "expo start",

    "reset-project": "node ./scripts/reset-project.js",

    "android": "expo start --android",

    "ios": "expo start --ios",

    "web": "expo start --web",

    "lint": "expo lint"

  },

  "dependencies": {

    "@expo/vector-icons": "^14.1.0",

    "@react-navigation/bottom-tabs": "^7.3.10",

    "@react-navigation/elements": "^2.3.8",

    "@react-navigation/native": "^7.1.6",

    "@tamagui/config": "^1.132.20",

    "expo": "~53.0.20",

    "expo-blur": "~14.1.5",

    "expo-constants": "~17.1.7",

    "expo-font": "~13.3.2",

    "expo-haptics": "~14.1.4",

    "expo-image": "~2.4.0",

    "expo-linking": "~7.1.7",

    "expo-router": "~5.1.4",

    "expo-splash-screen": "~0.30.10",

    "expo-status-bar": "~2.2.3",

    "expo-symbols": "~0.4.5",

    "expo-system-ui": "~5.0.10",

    "expo-web-browser": "~14.2.0",

    "react": "19.0.0",

    "react-dom": "19.0.0",

    "react-native": "0.79.5",

    "react-native-gesture-handler": "~2.24.0",

    "react-native-reanimated": "~3.17.4",

    "react-native-safe-area-context": "5.4.0",

    "react-native-screens": "~4.11.1",

    "react-native-svg": "^15.12.1",

    "react-native-web": "~0.20.0",

    "react-native-webview": "13.13.5",

    "tamagui": "^1.132.20"

  },

  "devDependencies": {

    "@babel/core": "^7.25.2",

    "@tamagui/babel-plugin": "^1.132.20",

    "@types/react": "~19.0.10",

    "eslint": "^9.25.0",

    "eslint-config-expo": "~9.2.0",

    "typescript": "~5.8.3"

  },

  "private": true

}

Tamagui.config.ts

import { config as configBase } from '@tamagui/config/v3'

import { createTamagui } from 'tamagui'

const config = createTamagui(configBase)

export type Conf = typeof config

declare module 'tamagui' {

  interface TamaguiCustomConfig extends Conf {}

}

export default config