nicotine\_level.dart

class NicotineLevel {

final DateTime date;

final double level;

NicotineLevel(this.date, this.level);

}

NicotineChartsPage.dart

import 'package:charts\_flutter/flutter.dart' as charts;

import 'package:flutter/material.dart';

import 'package:progetto/charts/nicotine\_level.dart'; // Assicurati che il percorso sia corretto

class NicotineChart extends StatelessWidget {

final List<charts.Series<NicotineLevel, DateTime>> seriesList;

final bool animate;

NicotineChart(this.seriesList, {this.animate = false});

@override

Widget build(BuildContext context) {

return charts.TimeSeriesChart(

seriesList,

animate: animate,

dateTimeFactory: const charts.LocalDateTimeFactory(),

);

}

}

NicotineTracker.dart

class NicotineTracker {

DateTime registrationDate;

Map<DateTime, int> cigarettesPerDay = {};

NicotineTracker(this.registrationDate);

void addCigarette() {

DateTime today = DateTime.now();

if (!cigarettesPerDay.containsKey(today)) {

cigarettesPerDay[today] = 0;

}

cigarettesPerDay[today] = cigarettesPerDay[today]! + 1;

}

int daysSinceRegistration() {

return DateTime.now().difference(registrationDate).inDays;

}

int calculateWeeklyThreshold(int daysSinceStart) {

int weeksSinceStart = (daysSinceStart / 7).floor();

return 5 - weeksSinceStart;

}

Map<DateTime, int> getCigarettesPerDay() {

return cigarettesPerDay;

}

}

plot\_creation.dart

import 'package:flutter/material.dart';

import 'package:charts\_flutter/flutter.dart' as charts;

// Define the NicotineLevel class

class NicotineLevel {

final DateTime date;

final double level;

NicotineLevel({required this.date, required this.level});

}

class NicotineChart extends StatelessWidget {

final List<charts.Series<NicotineLevel, DateTime>> seriesList;

final bool animate;

NicotineChart(this.seriesList, {this.animate = false});

@override

Widget build(BuildContext context) {

return charts.TimeSeriesChart(

seriesList,

animate: animate,

dateTimeFactory: const charts.LocalDateTimeFactory(),

);

}

// Method to create sample data for the chart

static List<charts.Series<NicotineLevel, DateTime>> createSampleData(List<NicotineLevel> data) {

return [

charts.Series<NicotineLevel, DateTime>(

id: 'Nicotine Level',

colorFn: (\_, \_\_) => charts.MaterialPalette.blue.shadeDefault,

domainFn: (NicotineLevel levels, \_) => levels.date,

measureFn: (NicotineLevel levels, \_) => levels.level,

data: data,

)

];

}

}