# Fitness Tracker Web API

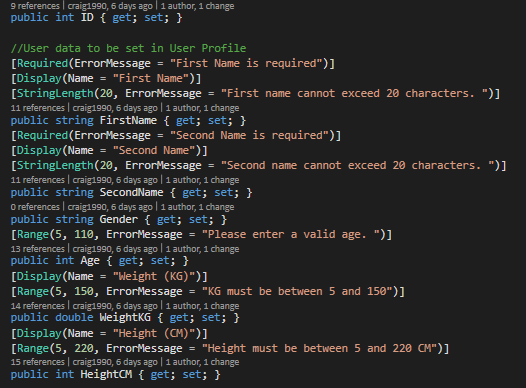
GitHub repository URI: <https://github.com/craig19/WebAPIFitnessTracker>

Swagger Endpoint: <http://fitnessapi-dev.eu-west-1.elasticbeanstalk.com/swagger/>

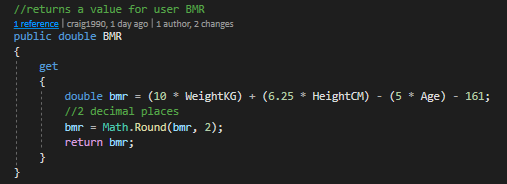
The Fitness Tracker Web API was setup to serve an Android app and its users, their details and their personal workout/exercise details.

The API allows the Android client to:

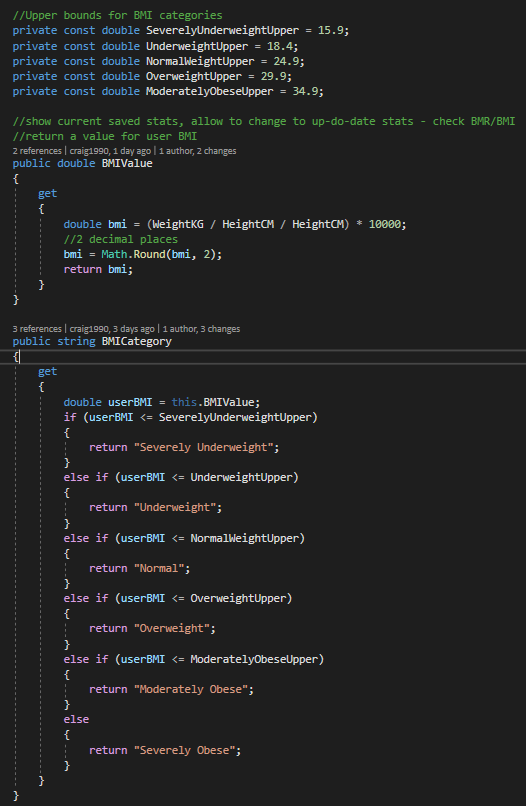
* Get/add/update/delete app Users and Users personal data (which will be done from a User Profile activity on the Android app). These details include First and Second Names, Age, Weight in KG and Height in CM.



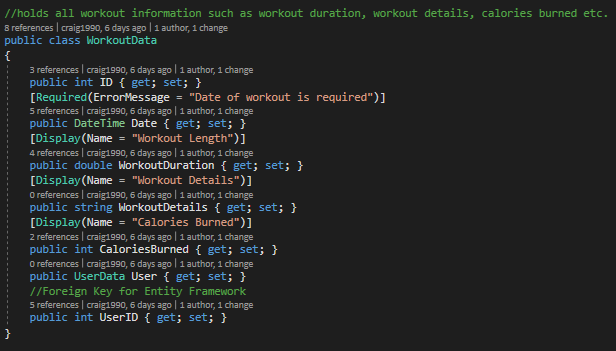
* Get a User’s Basal Metabolic Rate (BMR) from the User’s age, weight and height data, which is stored in the database. BMR is the number of calories required to keep your body functioning at rest; closely related to your body’s metabolism.



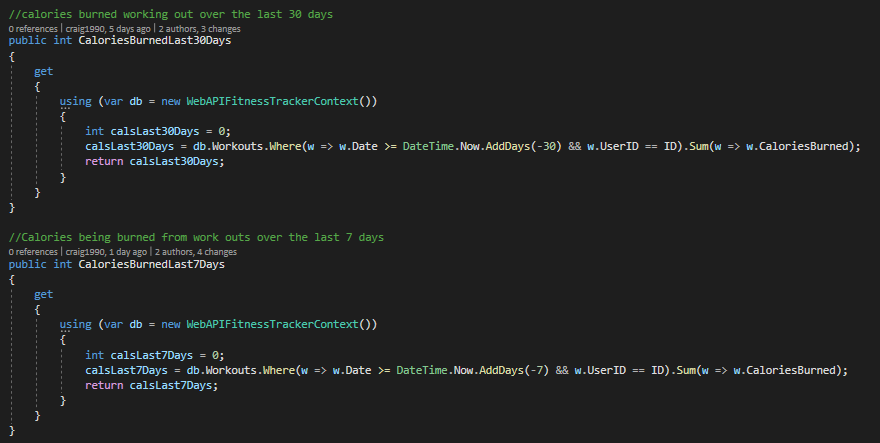
* Get a User’s Body Mass Index (BMI) from the User’s weight and height data which is stored in the database. BMI is a value derived from your weight and height to determine if you are a healthy weight e.g. what category you belong in: severely underweight, underweight, normal, overweight, obese.

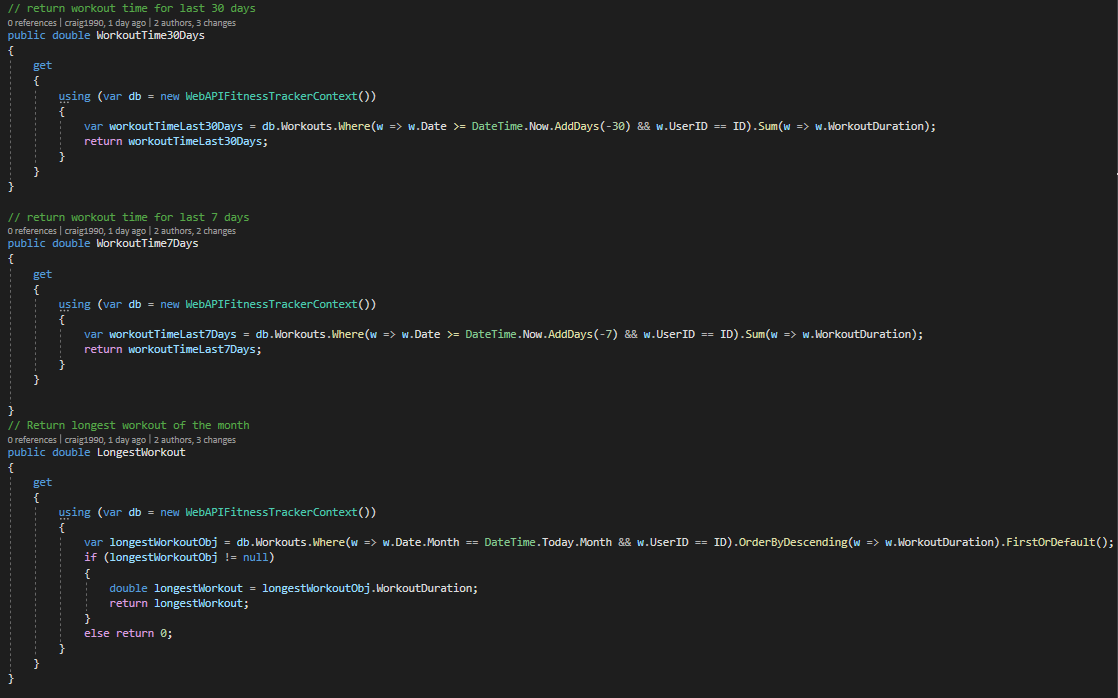


* Add a User’s personal workouts to their workout list; a workout includes data such as date of workout, workout duration, calories burned during the workout (easily attainable nowadays from smart watches), and workout details (exercises completed etc.). This will allow the app to show a list of a User’s past workouts to them, to allow User’s to track their time working out, their calories burned working out and also keep track of their workout details such as body parts worked on and exercises completed etc.

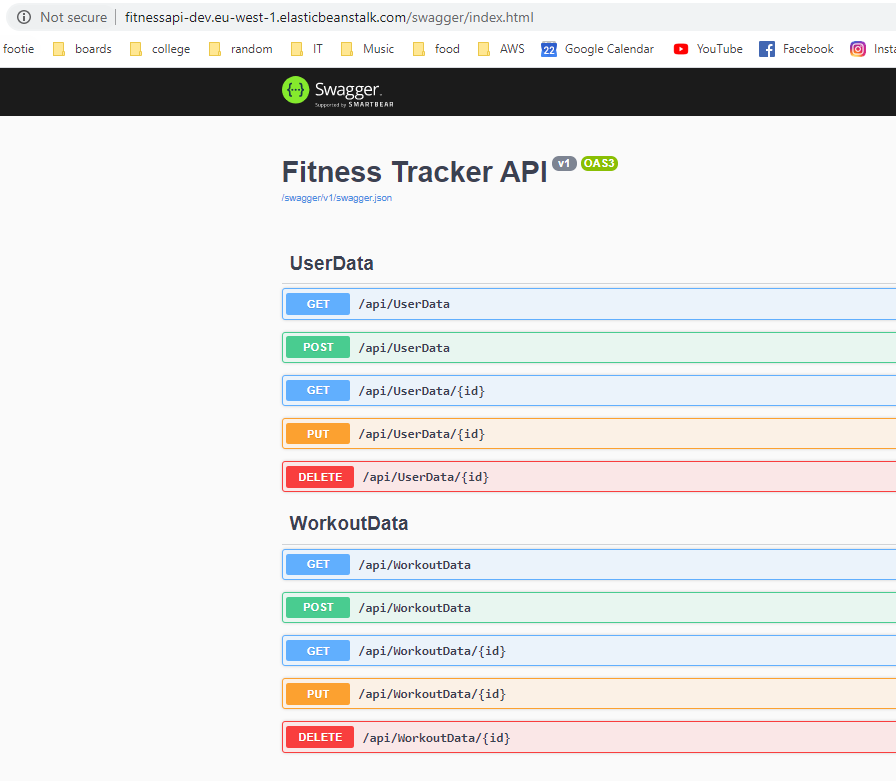


* Get a User’s workout duration and calorie statistics e.g. total calories burned over the last 7 and 30 days and total time working out over the last 7 and 30 days, longest workout of the month.

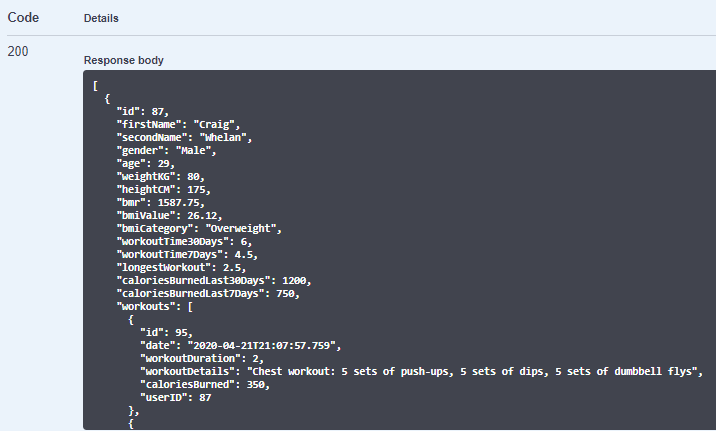


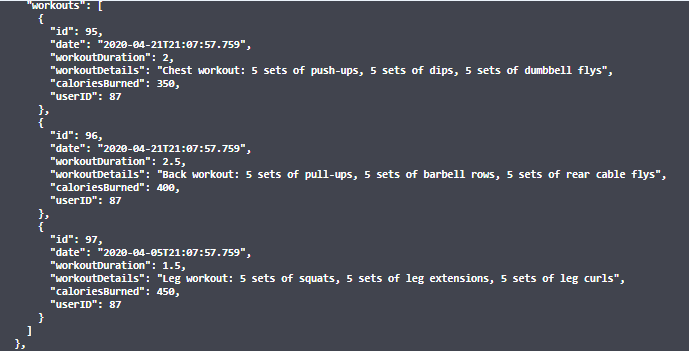


The Swagger page showing all API endpoints to facilitate the above. The project video shows a running Swagger test; adding a user with initial details, adding workouts to the users list of workouts and getting that users details back including all additional stats such as BMR, BMI and calorie/workout length stats.

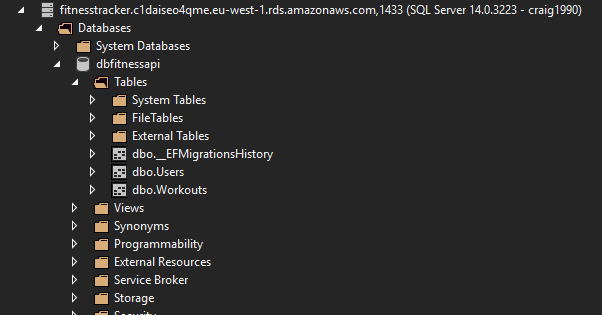


A full User object

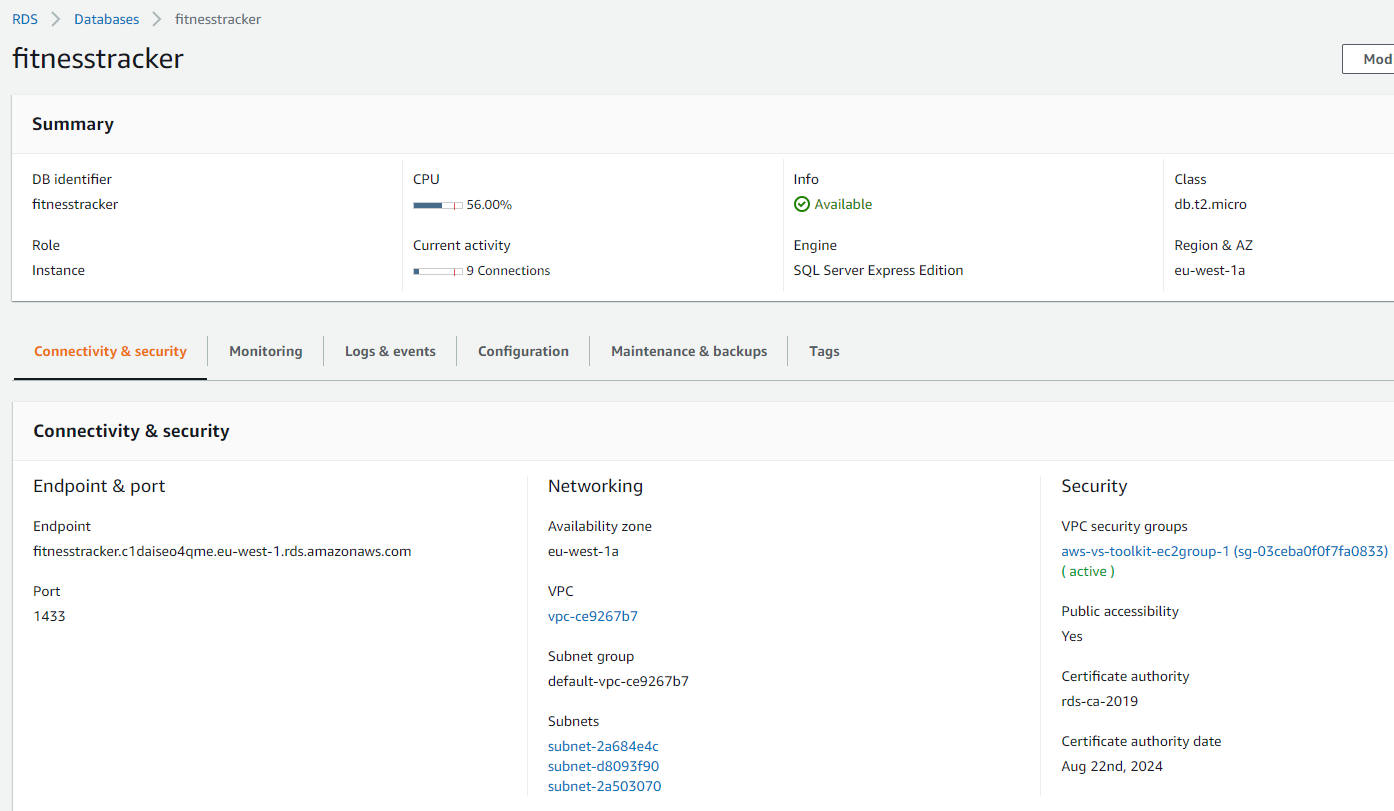




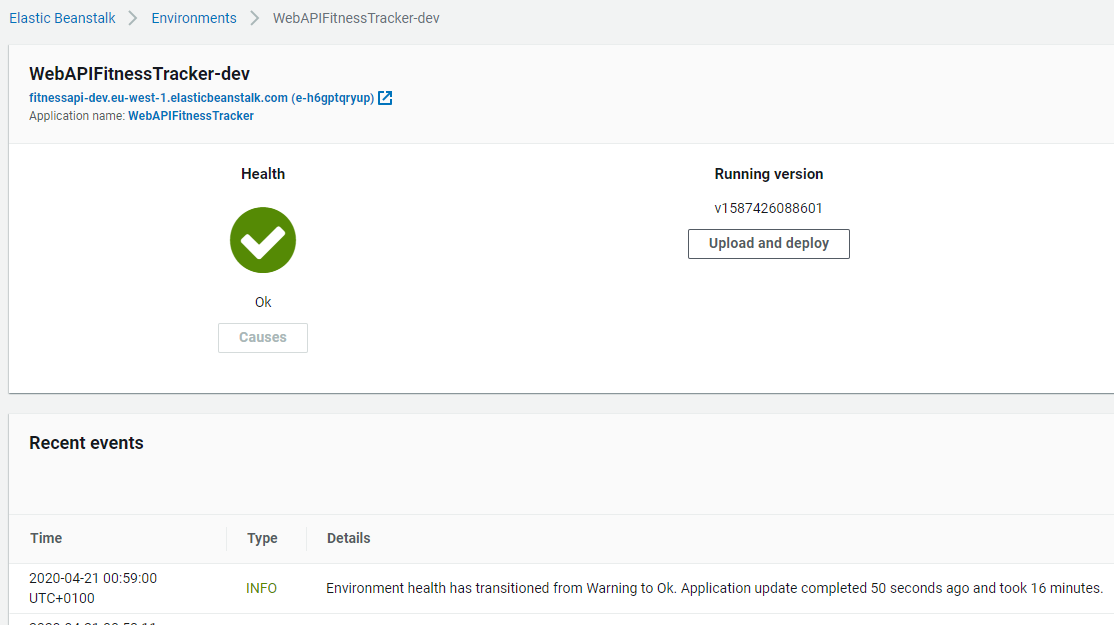
2 database tables 1 for Users and one for Workouts. A user has a list of workouts and entity framework uses userID as foreign key for map workouts to users.

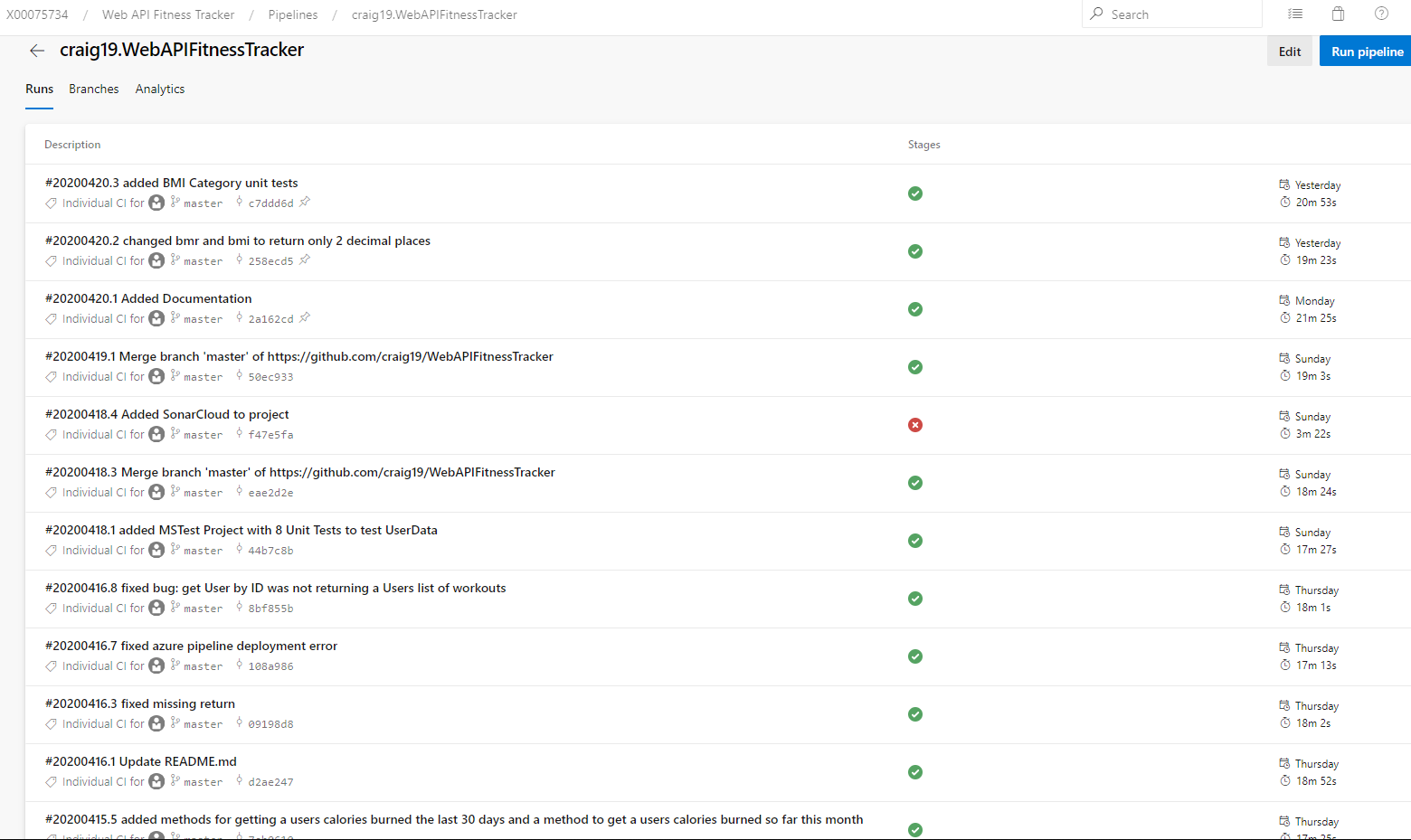


Database is hosted on AWS RDS



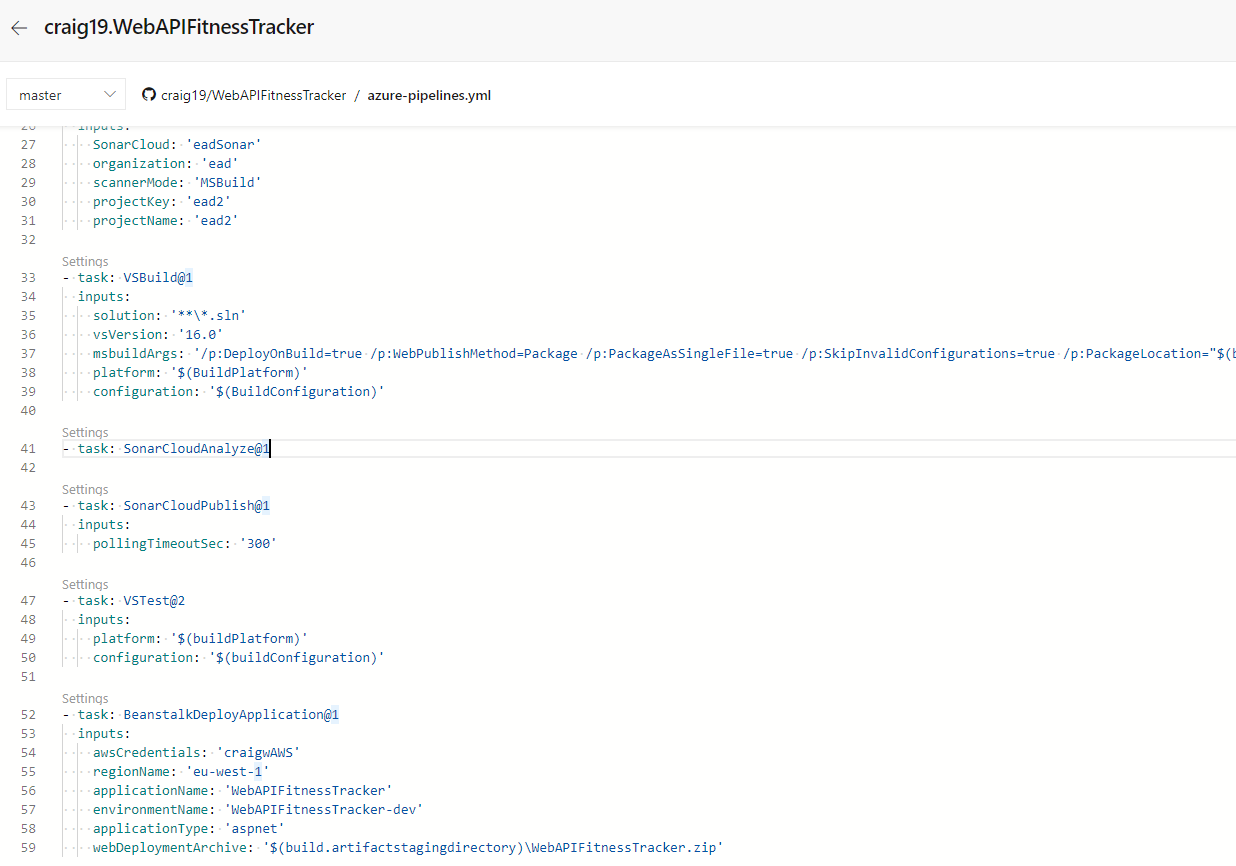
RESTful web service is hosted on AWS Elastic Beanstalk via an Azure DevOps pipeline which runs our unit tests and static code analysis.



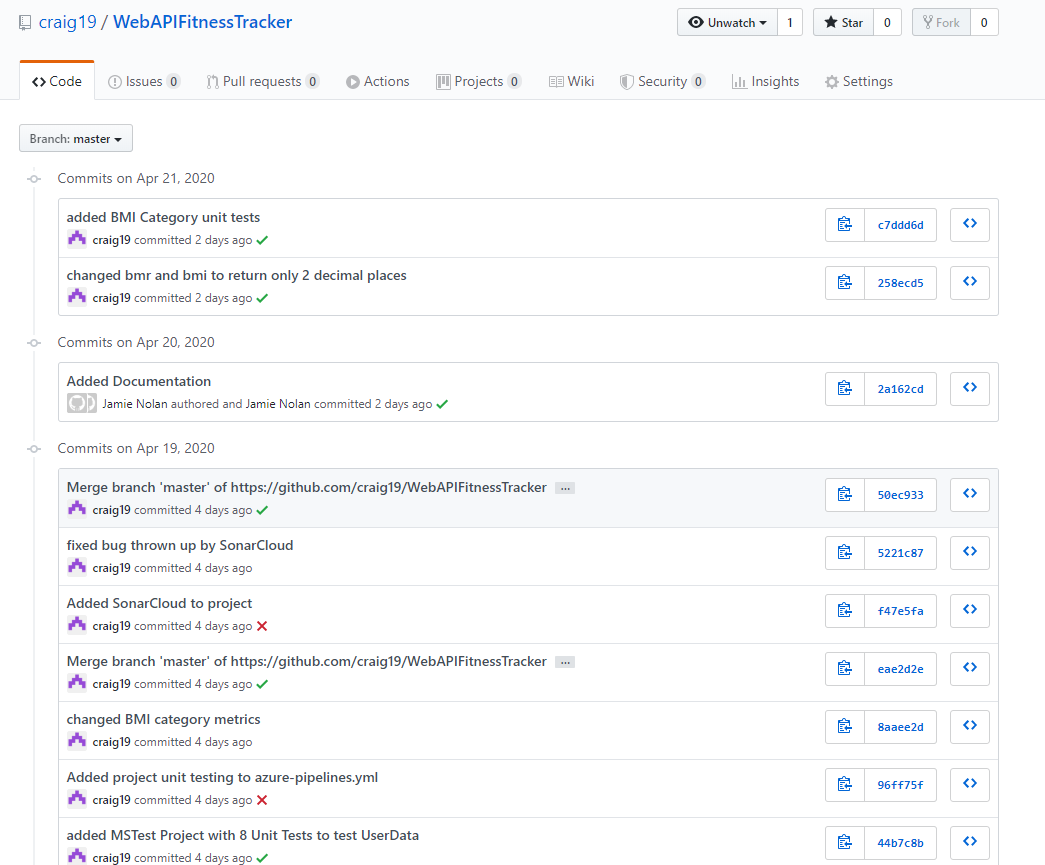


Full .yml file available on web API Git repo or you can view the public Azure DevOps project:

<https://dev.azure.com/X00075734/Web%20API%20Fitness%20Tracker>



GitHib has been fully utilized. All commits to the Web API repo kick-off an Azure pipeline build which analyses our code with SonarCloud and runs our unit tests. If these unit tests pass it will deploy to our development env on AWS Elastic Beanstalk.



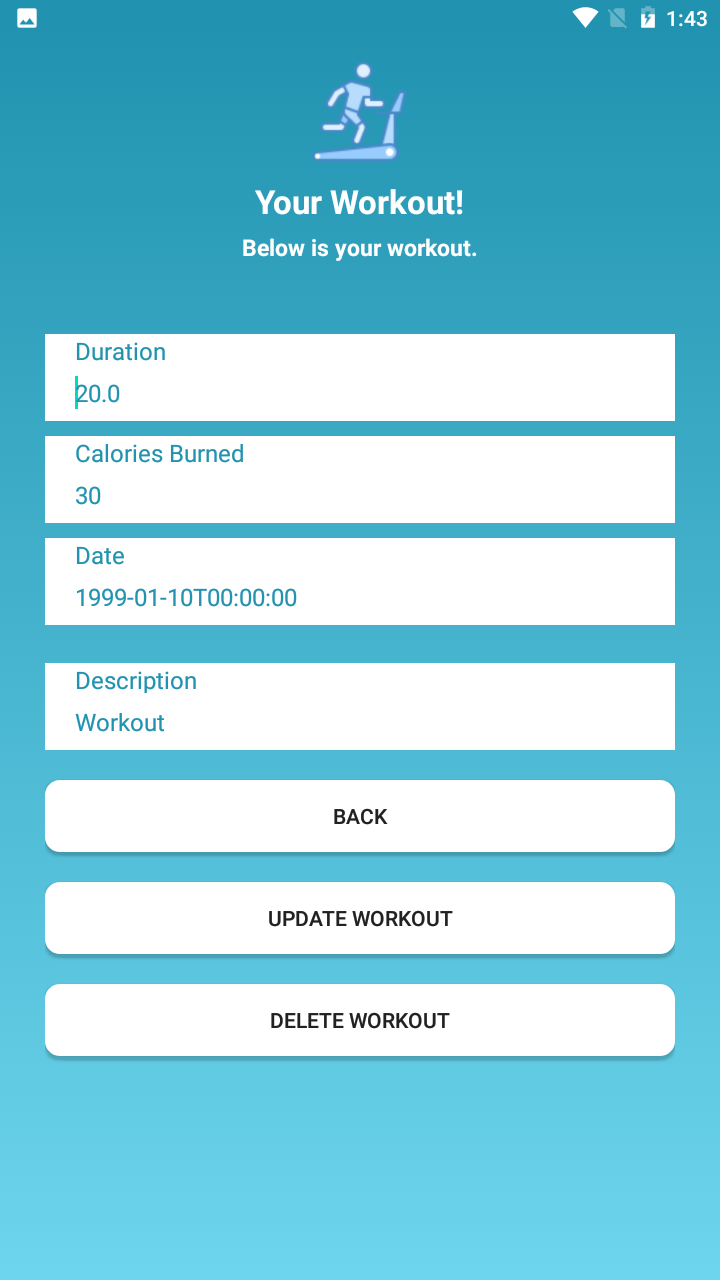
Unit tests (full list available on web API Git repo)

|  |  |
| --- | --- |
|  | Fitness Tracker |
| Fitness TrackerAndroid App | |

|  |
| --- |
| Fitness Tracker Android App  Android Github: <https://github.com/jamieNolanDEV/FitnessTrackerAndroid>  What is Fitness Tracker?  Fitness Tracker is an to view your BMI, view your BMR, add a workout, view previous workouts and get a breakdown of stats  Technologies used:  Okhttp3  OkHTTP3 is an open source library designed for android to be an efficient HTTP client.  Okhttp3 is used for our get, post, put and delete methods entirely.  Java.  Used instead of Kotlin as JAVA has been thought.  Android Utilities used:   * Card view – Used to display items in a nice fashion * Recycler View – Create instances of workouts and populate a array list in which the recycler view can display all our workouts. * Shared Preferences – Used to Verify that a user does not exist or exist. Store strings variables e.g. user id locally on android so it can be retrieved even when application is closed. Used to check if it’s a users first time using the app.   Testing – Espresso.  Espresso is used to end to end test our app. From view BMI,BMR, View User Stats, add workouts and view workouts.  Screenshots of app from start to finish  :  Adding in a users details:  First Screen if userid does not exist, this is the screen you will see.  Contains Validation if wrong information is added.      Main page after adding a user:  Main Page utilizing image views, icons and text so a user can move throughout the app    View BMR:  View BMR page represents a get request based on the user ID and populate the field below.  Get request is using okhttp, TextView is then populated    View BMI:  View BMI page represents a get request based on the user ID and populate the field below.  Retrieved using okhttp with userId as parm and set textviews to variables.    User Details:  User details : on this page we can update or delete our user  Update: Put request taking all previous parms, Delete – Delete request - okhttp  User Details:  User details : on this page we can update or delete our user.    User Stats:  User stats populates the textview with get request using okhttp for the userId it has.  Retreived using okhttp get request and populates all the textviews with response.    Add a workout page:  Page that will add a workout with editexts that will send this information to the service. Using put in okhttp  Added Workout details:  Added workout with details populated  Workout List, populated with query.  Query being workouts for that userId. Workouts populated via arraylist and viewed with RecyclerView. |

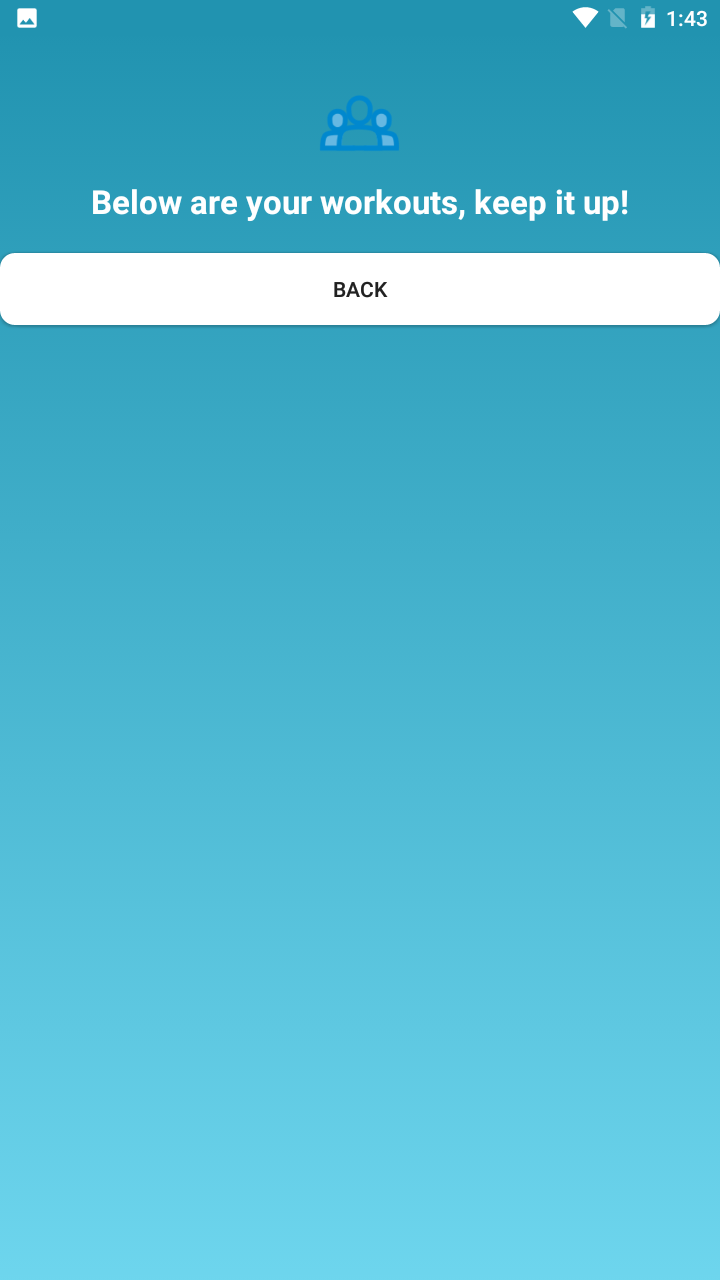
View Workout, can update and delete from here:

Can now view edit and delete this workout and show the information now exists



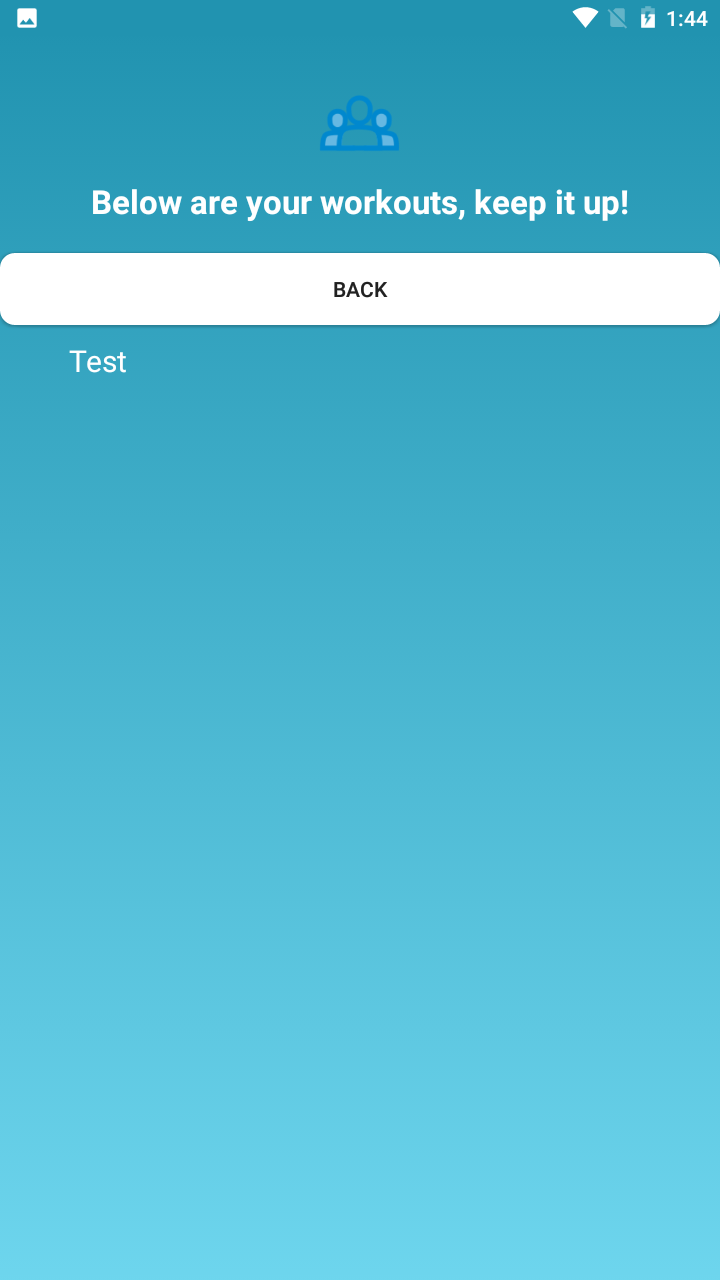
After Deleting this workout:

Shows that the workout has been deleted and is nowhere available on the app anymore.



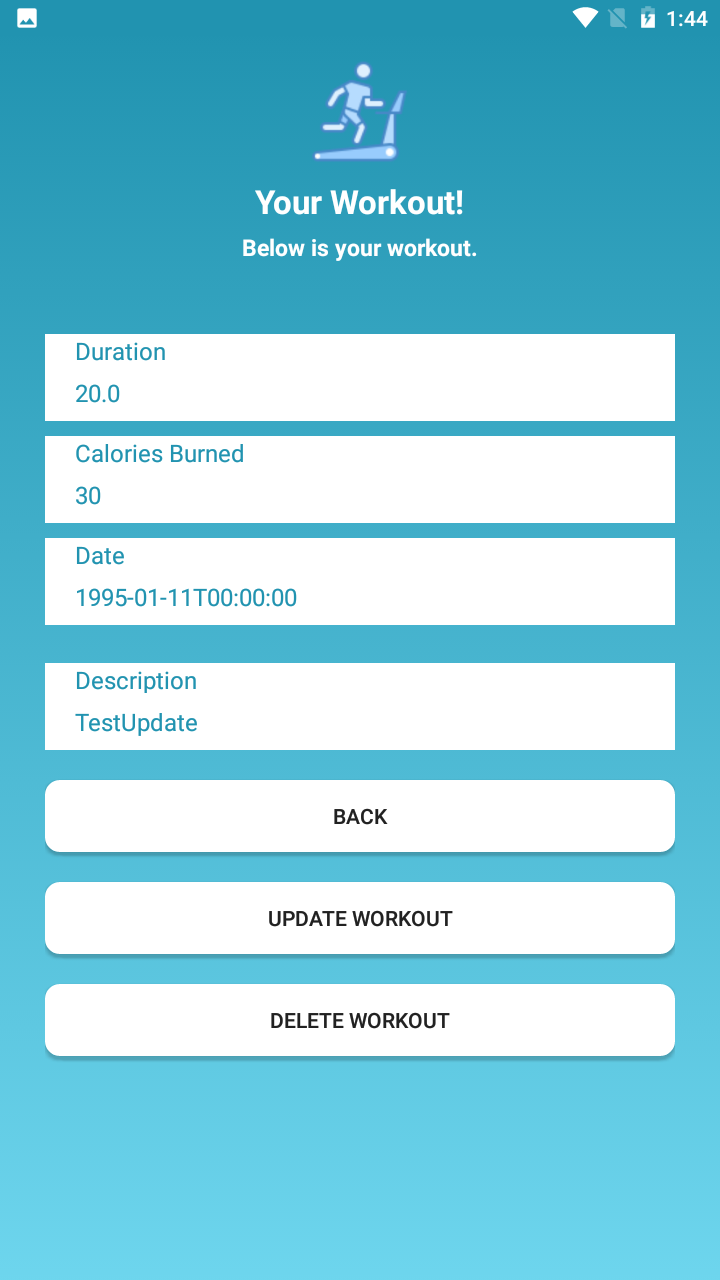
Add new workout test:

Shows that a new workout with the desc Test has been added. Using RecyclerView to display



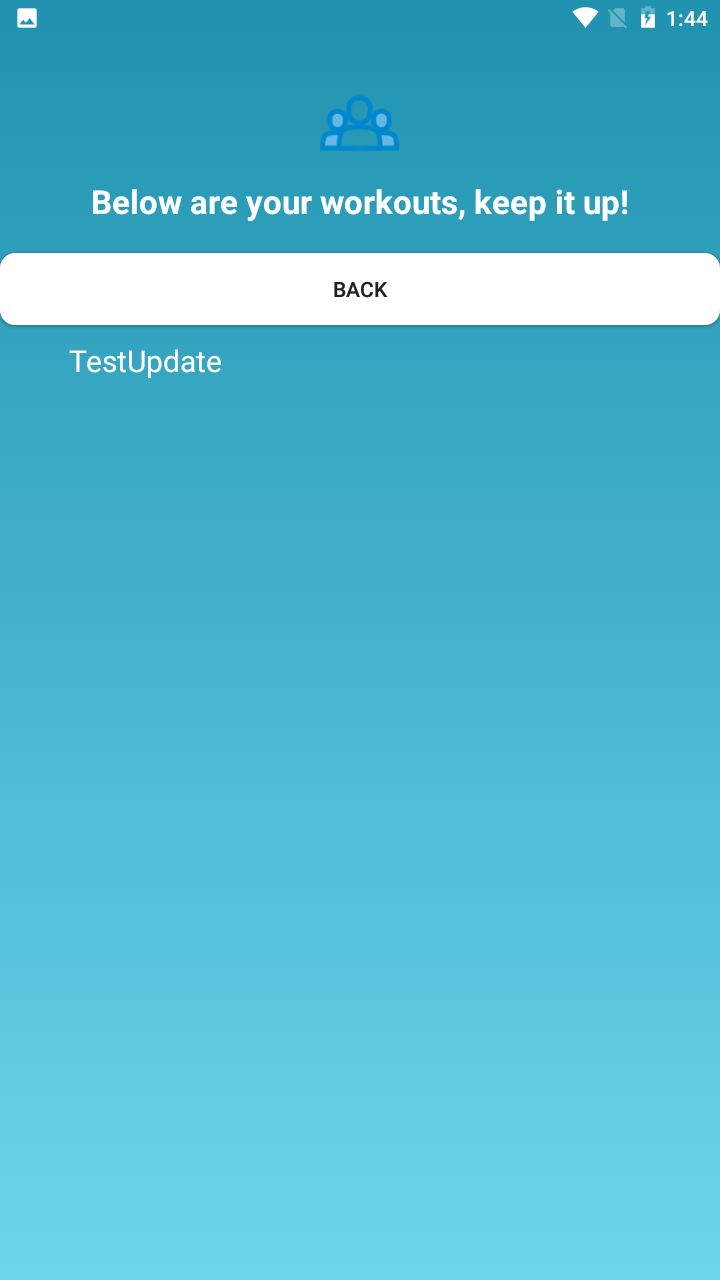
Edit this workout:

Fields are editable to allow the user to update their workout.



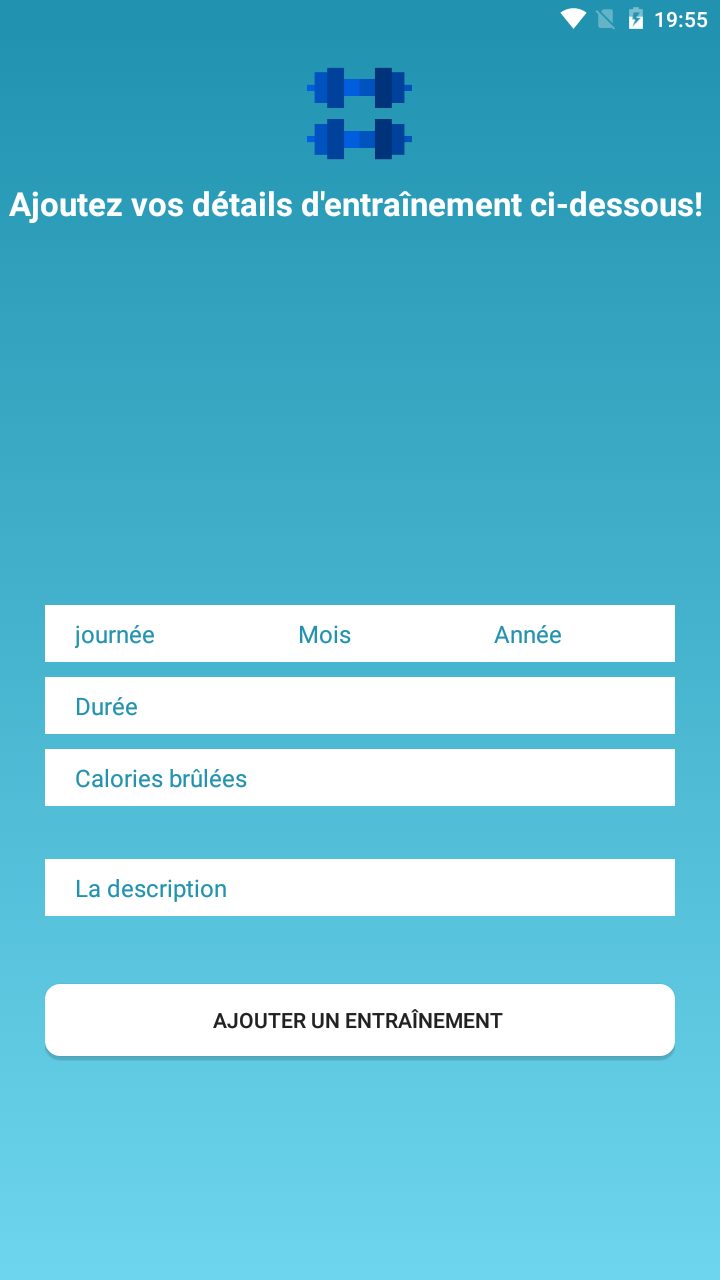
Updated details added:

Verification that this information has been updated: Test has became TestUpdate. - .put method

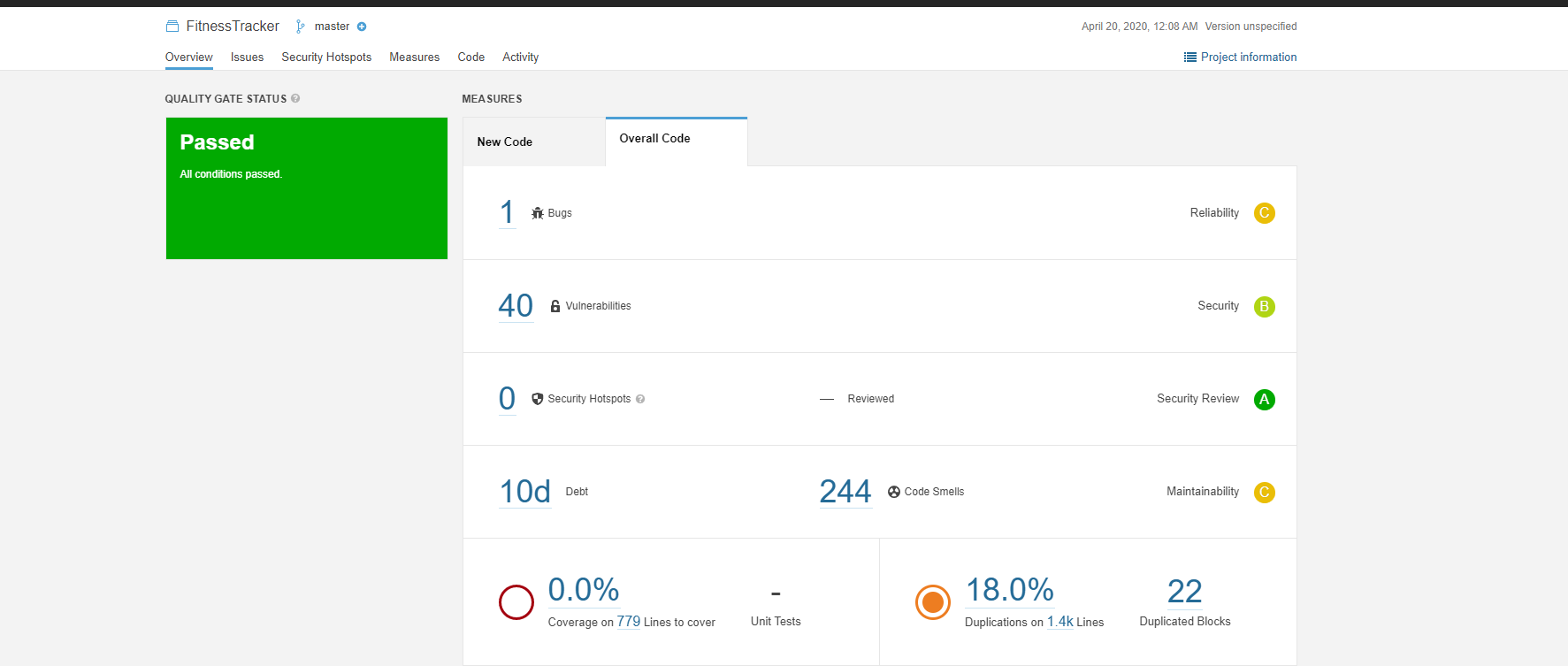


Example of Localization in french:

Using strings.xml we can create a localized version of all our strings: Example being our add workout page in french used because the emulator is in french.



Testing – SonarQube



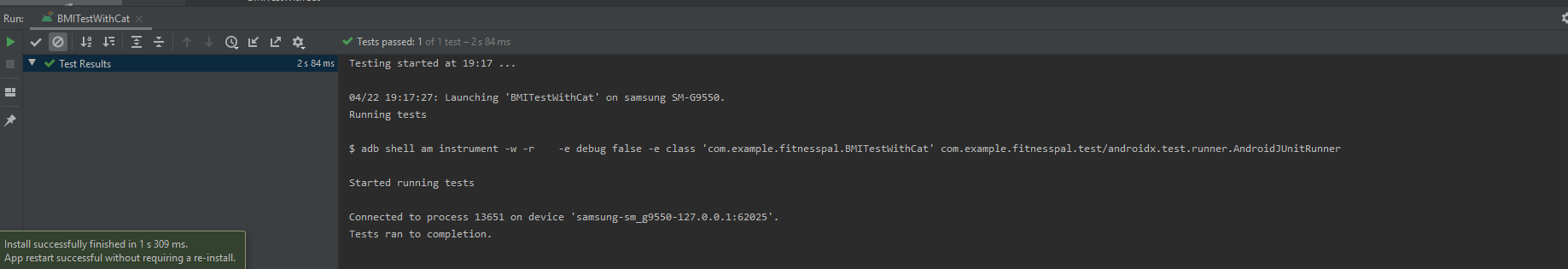
Testing – Espresso

BMR Test – Create User and View BMR is correct Value.



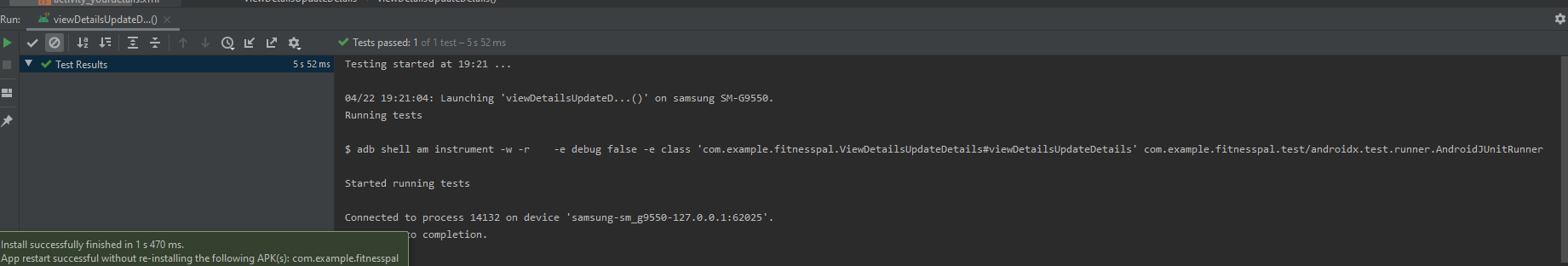
BMI

Test – Create User and View BMI is correct Value and correct category.



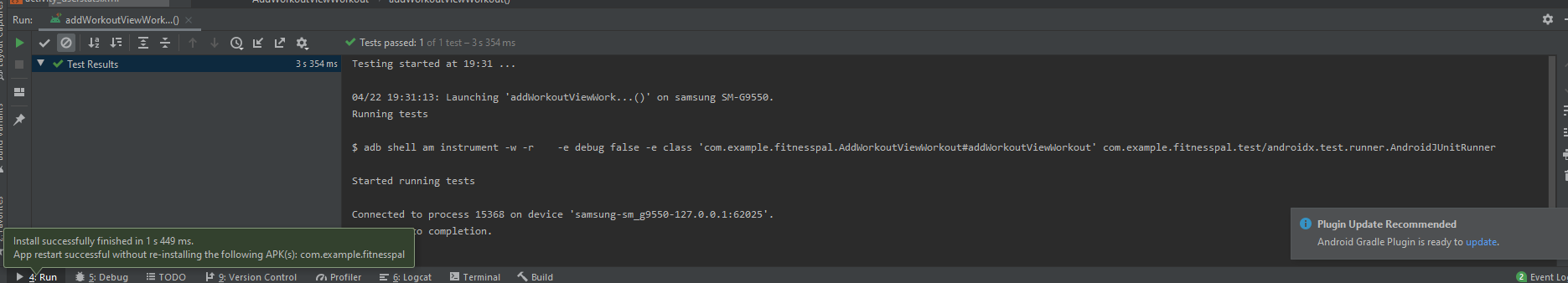
Details

Test – Create a user, view First Name, update First Name assert new name is set.



Workout

Test – Add Workout, View Newly Created Workout



View Workout

Test – Add Workout, View Longest workout in userstats

