

CYBR8470 Secure Web Applications Development

Fall Semester 2021

Class Project:
Mobile and Web Application for
Amateur Radio Emergency Services
'AERS'



Bob Wilmes AGØLF
December 16, 2021

Background of the AERS project



The initial goal of the project is to support data collection by ham radio operators operating as a communications network to recover from natural and man-made disasters. In Omaha, the Amateur Radio Emergency Service (ARES) is affiliated with the Aksarben chapter of the American Red Cross. ARES is organized in the US by the Amateur Radio Relay League, a 501c3 charity which promotes amateur radio use.



The project targeted a mobile application using the Dart programming language running under the Flutter framework on iOS, Android and Chrome web browsers. Dart was developed along with the Flutter framework by Google. It generates native machine code for ARM32, ARM64, Intel x86_64 CPU's and cross compiles to JavaScript and Web Assembler language.

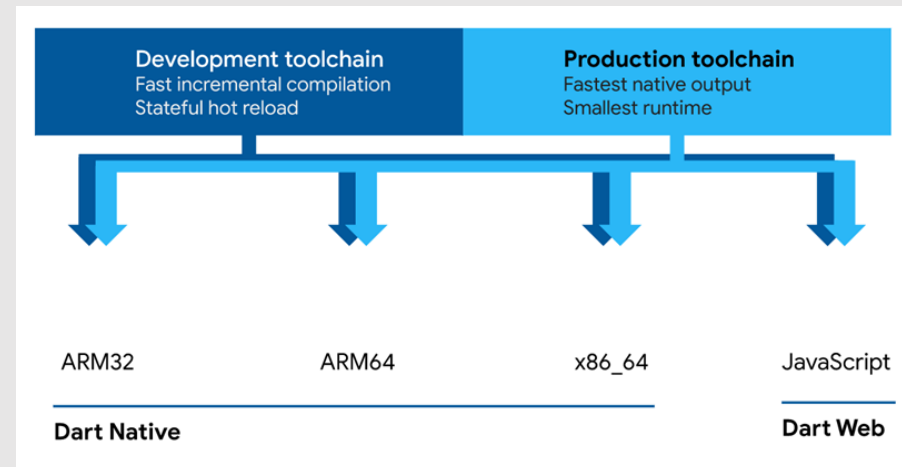
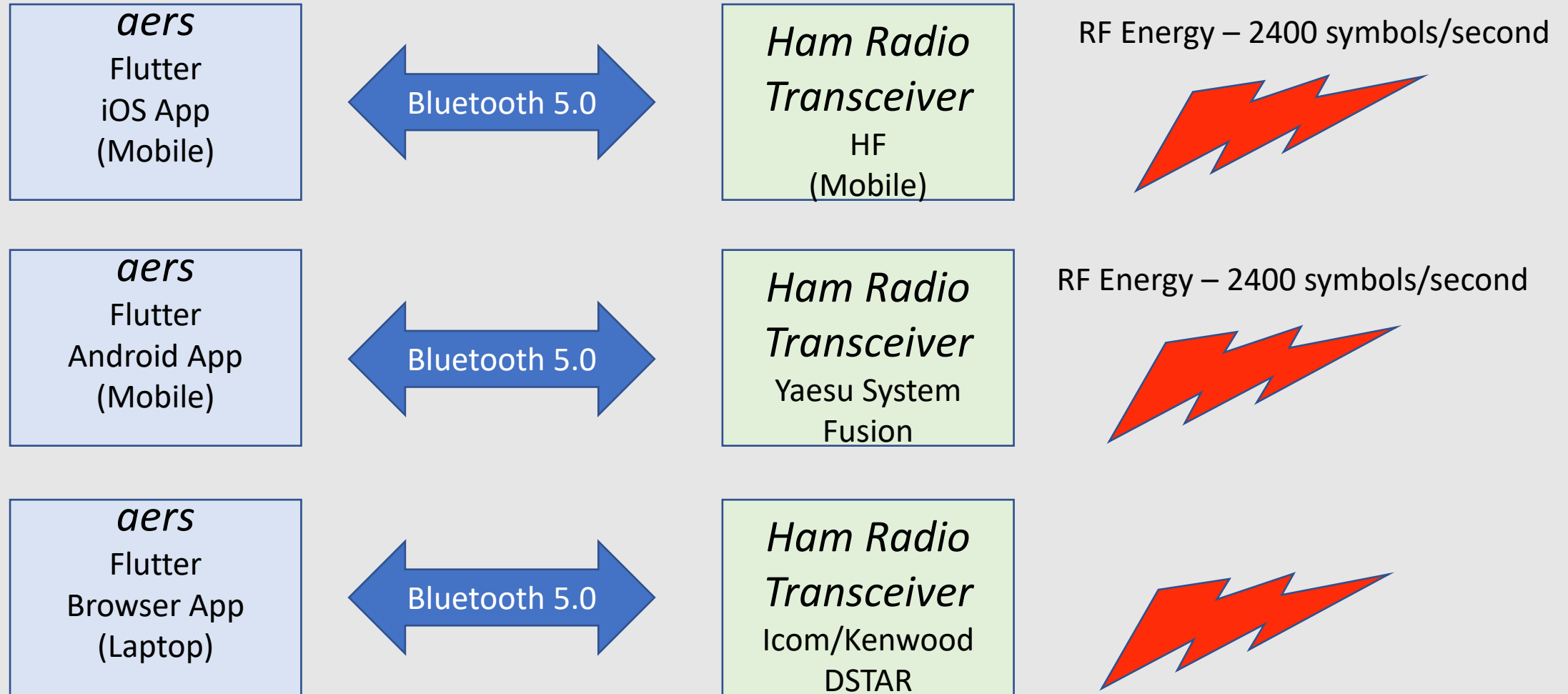
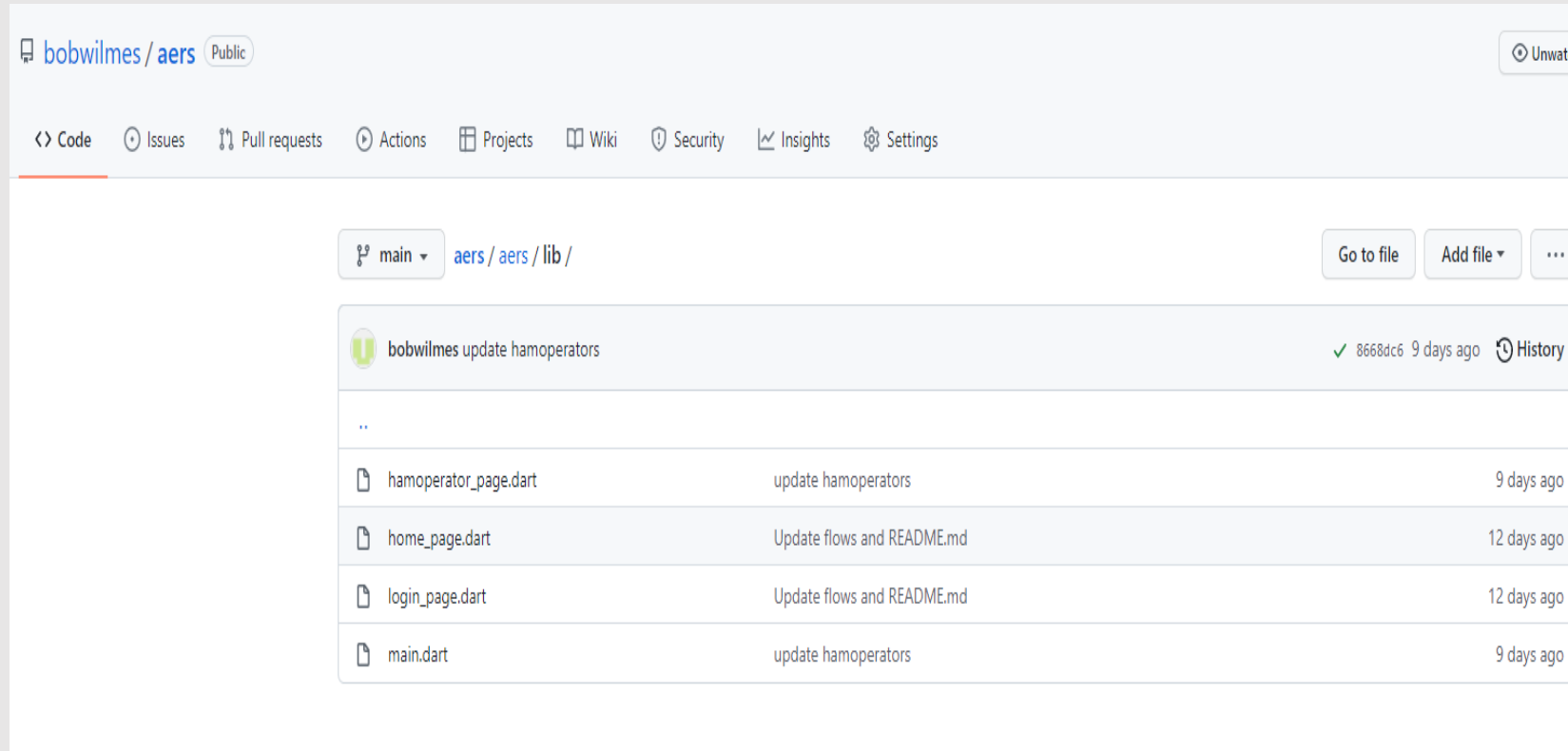


Figure 1 - Aers Architecture Block Diagram



AERS – the incomplete app

As I became deeply involved in the Ham Radio application it became evident that as I added additional screens the complexity exceeded my skill level in Dart and Flutter. I spent literally a hundred hours watching flutter development videos.



Incomplete repository:
<https://github.com/bobwilmes/aers.git>



Flutter 2.0 March 2021 breaks the world!

The addition on the null safety to Flutter 2.0 in March 2021 caused an issue for many existing code projects in github.

Flutter's version of Lint (“\$ dart analyze .”) can report on the variances but does not mitigate them.

In addition, Flutter guides and references published prior to March 2021 do not cover the issues at all.

What's New in Flutter 2

Flutter web and Null Safety move to stable, Flutter desktop moves to beta and so much more!



Chris Sells

Follow



Mar 3 · 19 min read



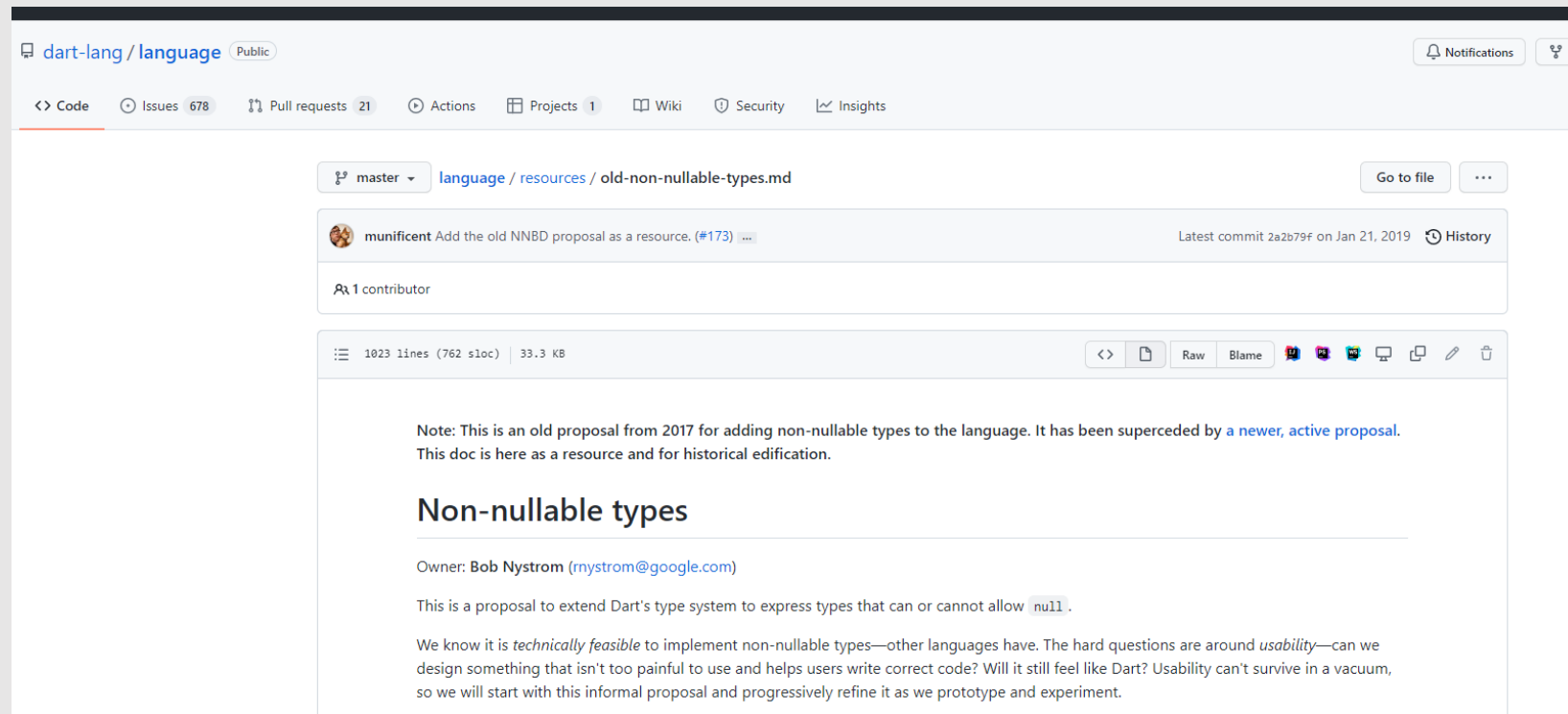
Today, we're pleased to announce the release of Flutter 2. It's been a little more than two years since the Flutter 1.0 release, but in that short time, we've closed 24,541 issues and merged 17,039 PRs from 765 contributors. Just since the Flutter 1.22 release in September, we've closed 5807 issues and merged 4091 PRs from 298 contributors. Special thanks go out to our volunteer contributors who generously give their spare time to improve the Flutter project. The top volunteer contributors for the Flutter 2 release were [xu-baolin](#) with 46 PRs, [a14n](#) with 32 PRs that focused on bringing Flutter to null safety, and [hamdikahloun](#) with 20 PRs that improved a number of the Flutter plugins. But it's not just coders that contribute to the Flutter project; a great set of volunteer PR reviewers were also responsible for reviewing 1525 PRs, including [hamdikahloun](#) (again!), [CareF](#) and [YazeedAlKhalaf](#) (who's only 16!). Flutter is truly a community effort and we couldn't have gotten to version 2 without the issue raisers, PR contributors, and code reviewers. This release is for all of you.

<https://medium.com/flutter/whats-new-in-flutter-2-0-fe8e95ecc65>



Dart's non-nullable datatypes

One of the difficulties learning flutter and Dart is the changes still being made to the language. In 2017, Bob Nystrom from Google started adding features that prohibited references to functions that previously could have passed or returned values as null. This is explained here but has broken a lot of code released prior to Flutter 2.2



<https://github.com/dart-lang/language/blob/master/resources/old-non-nullable-types.md>



Completing the World Time application

One of the maxims of the software world is working code carries 90% of the weight of a project. I decided to simplify my project. My plans for the ham radio project became much more complex.

As I didn't understand async code and futures required to communicate externally off the mobile device.

I chose to base my project on the "Flutter-Beginners-Tutorial" github project and YouTube videos.

<https://github.com/iamshaunjp/flutter-beginners-tutorial>

<https://www.youtube.com/watch?v=1ukSR1GRtMU&list=PL4cUxeGkcC9jLYyp2Aoh6hcWuxFDX6PBJ>

As I watched the videos, I manually entered the code from the video character by character. This was so I could utilize Flutter's instantaneous compilation feature and painfully learn Dart syntax.



The challenge was the videos and code were written for Flutter 1.0 and haven't been upgraded since 2019.

My project became to modernize the World Time application to run under non null code under Flutter 2

Completed Flutter 2 World Time repository:

<https://github.com/bobwilmes/worldtime.git>



World Time app showing the changes in the background and screens

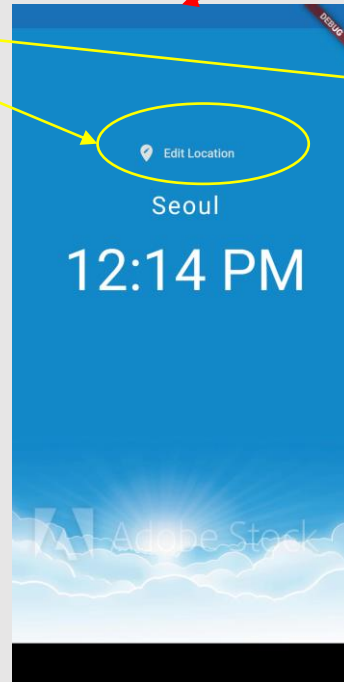
Touch Edit Location
Icon to change locations

Note: slightly changed color on upper app screen bar between day and night

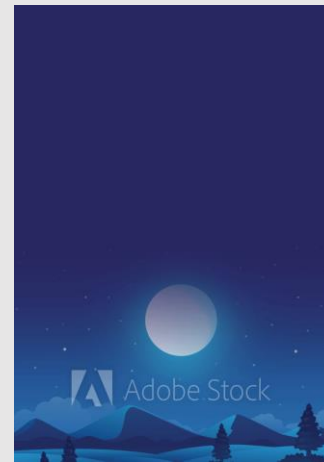
Touch to return



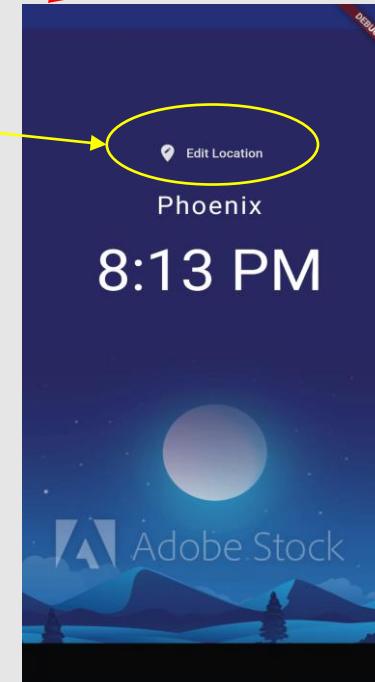
Daytime
Background



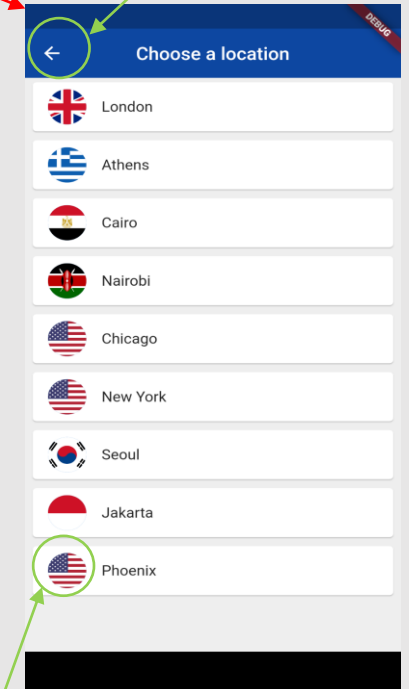
Stretched Background
Time and Location
With Edit Location
Control



Nighttime
Background



Stretched Background
Time and Location
With Edit Location
Control



Choose Location
Screen with
Cities and Flags
Buttons

Touch Flag Icons to select city

What I Learned from World Time for Flutter 2

The World Time application uses several functions which were required of the project. First, an initial loading page is created while a https call is made to a JSON REST web service provided by WorldTimeAPI.org

Because flutter code runs asynchronously a rendezvous is made between the https library call and an a callback using the “future” option. The call returns a list of parameters which is parsed to JSON and the parameters are placed in memory using a map object.

The home page is placed on the stack and shows the current time and location. Background shading is applied depending on local time of day.

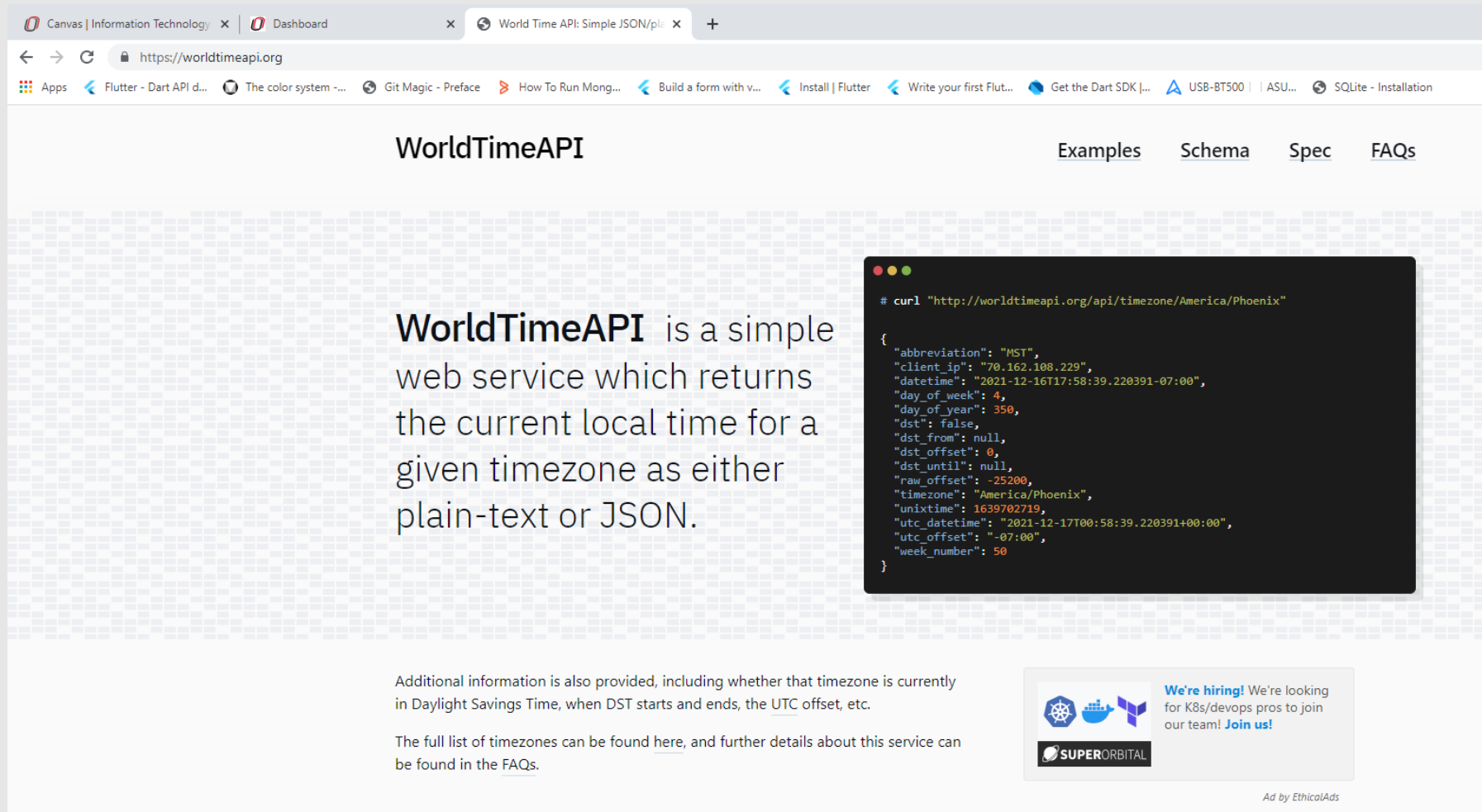
Second, an edit location button loads a screen on the stack, which allows the user to choose alternate locations around the world. The icons for the locations are created dynamically at run time from list. When a icon is clicked, the location screen is popped off the flutter stack and the parameters are passed back using a map.

Third, I learned how to separate the organization of code in the project into separate subdirectories depending on the screen in the application. This made the debugging process clear, given the Flutter stack use to transfer screens.



WorldTimeAPI.org

This is the https asynchronous request to obtain the current for a geographic location



The screenshot shows a web browser window with the URL `https://worldtimeapi.org`. The page title is "WorldTimeAPI" and it has navigation links for "Examples", "Schema", "Spec", and "FAQs". The main content area features a large text block stating: "WorldTimeAPI is a simple web service which returns the current local time for a given timezone as either plain-text or JSON." To the right of this text is a dark terminal window showing a `curl` command and its output. The command is `curl "http://worldtimeapi.org/api/timezone/America/Phoenix"` and the output is a JSON object containing various time-related fields. Below the main text, there is additional information about DST and UTC offset, and a link to the full list of timezones. At the bottom right, there is a "SUPERORBITAL" logo and a hiring notice.

WorldTimeAPI

Examples Schema Spec FAQs

WorldTimeAPI is a simple web service which returns the current local time for a given timezone as either plain-text or JSON.

```
# curl "http://worldtimeapi.org/api/timezone/America/Phoenix"
{
  "abbreviation": "MST",
  "client_ip": "70.162.108.229",
  "datetime": "2021-12-16T17:58:39.220391-07:00",
  "day_of_week": 4,
  "day_of_year": 350,
  "dst": false,
  "dst_from": null,
  "dst_offset": 0,
  "dst_until": null,
  "raw_offset": -25200,
  "timezone": "America/Phoenix",
  "unixtime": 1639702719,
  "utc_datetime": "2021-12-17T00:58:39.220391+00:00",
  "utc_offset": "-07:00",
  "week_number": 50
}
```

Additional information is also provided, including whether that timezone is currently in Daylight Savings Time, when DST starts and ends, the UTC offset, etc.

The full list of timezones can be found [here](#), and further details about this service can be found in the [FAQs](#).

SUPERORBITAL

We're hiring! We're looking for K8s/devops pros to join our team! [Join us!](#)

Ad by EthicalAds

Futures for the Project

I used the Android Studio to complete the World Time project, but I plan to return to the Visual Studio Flutter plug-ins to finish the emergency radio project.

The Flutter plug-in for Bluetooth supports streaming Bluetooth connections but I need to build a mini-project to determine Bluetooth Low Energy (BLE) support which doesn't require device pairing.

Add the SQLite database support for the project. Connect a bridge to the Firestore Google supported database for initial data load on the mobile device. This will enable loading reference data when connectivity to the Internet is not available.

Investigate taking pictures from Flutter and local storage.

Create forms for data entry and validation of user information.

Use the ARES Field Operations manual for additional reference information.

Build a testing/CI framework on Github



References

Flutter for Dummies
by Barry Burd, PhD
Copyright 2020 John Wiley & Sons
ISBN 978-1119612582

Flutter in Action
by Eric Windmill
Copyright 2020 Manning Publications
ISBN 978-1617296147

Flutter Complete Reference
by Alberto Miola
Copyright <none listed>
ISBN 979-8691939952

Amateur Radio Emergency Service®
ARES® Field Resource Manual
Copyright © 2005-2021
American Radio Relay League, Inc.
ISBN 978-0872595439

Completed repository:
<https://github.com/bobwilmes/worldtime.git>

Incomplete repository:
<https://github.com/bobwilmes/aers.git>

World Time API: <https://worldtimeapi.org>

