

# Syncthing Monitor

## Technical Report & Documentation

Programming for Mobile Devices

2021-05-16

Dan Cojocaru

Student, Subgroup 1, Year 2

PhD Liviu Octavian Mafteiu-Scai

Lecturer

West University of Timișoara

# Abstract

This report presents the current features and functionality of the proposed *Syncthing Monitor* application, detailing the achieved and remaining goals of the application, the intended users, a comparison with other potential options, a user manual for the current features and finally a brief exploration into the technical details.

## Goal

The goal of *Syncthing Monitor* is to facilitate the management of multiple Syncthing instances located on multiple computers, all from one application.

This is achieved by allowing the adding of multiple servers, and then being able to monitor and manage each of them from one convenient place.

For the moment, the management of the servers is limited in functionality, presenting a further goal of expanding this functionality.

## Users

The target of the application are mainly the “prosumer” (professional consumer), users who have a little more than beginner knowledge of the system and who use more than just their personal computers as synchronization targets. However, the application is designed in such a way that it should be easy for casual consumers to conveniently use the application as well.

While some professional user features will be added, they are not be the focus of this application, as professional users tend to use technologies such as SSH Tunnelling in order to manually manage and configure the machines, therefore not needing applications to do such work. As such, professional users will currently not find a great appeal in the application. Some basic monitoring, however, is likely still more convenient to be done from the application rather than manually.

# Introduction

Syncthing is a free, open-source peer-to-peer file synchronization application available for Windows, Mac, Linux, Android, Solaris, Darwin, and BSD. It can sync files between devices on a local network, or between remote devices over the Internet. Data security and data safety are built into the design of the software.<sup>1</sup>

Syncthing is installed on each computer where synchronization is desired. This leads to a problem because each computer has its own version of the Syncthing software with different settings. Making a change across the entire network of synced devices becomes a difficult task.

*Syncthing Monitor* will be designed to fulfil multiple purposes:

- monitoring each server in order to observe if synchronization is proceeding as expected
- changing server settings on one, multiple or all interconnected servers
- remotely pause synchronization or restart a problematic server
- exclude certain files from synchronization

For now, *Syncthing Monitor* completely achieves the first and third purposes, with work left to be done on the second and forth.

---

<sup>1</sup> <https://en.wikipedia.org/wiki/Syncthing>

## State of the Art

At the moment, there are no applications fulfilling the goals of this application.

Syncthing GTK<sup>2</sup> is a Linux application that uses similar techniques but only to monitor in a graphical environment the server running on the local machine. It does not provide monitoring of multiple machines.

Syncthing<sup>3</sup> – the Android application – provides a Syncthing server running on an Android device, but is otherwise not fulfilling any of the proposed goals.

## Author contribution

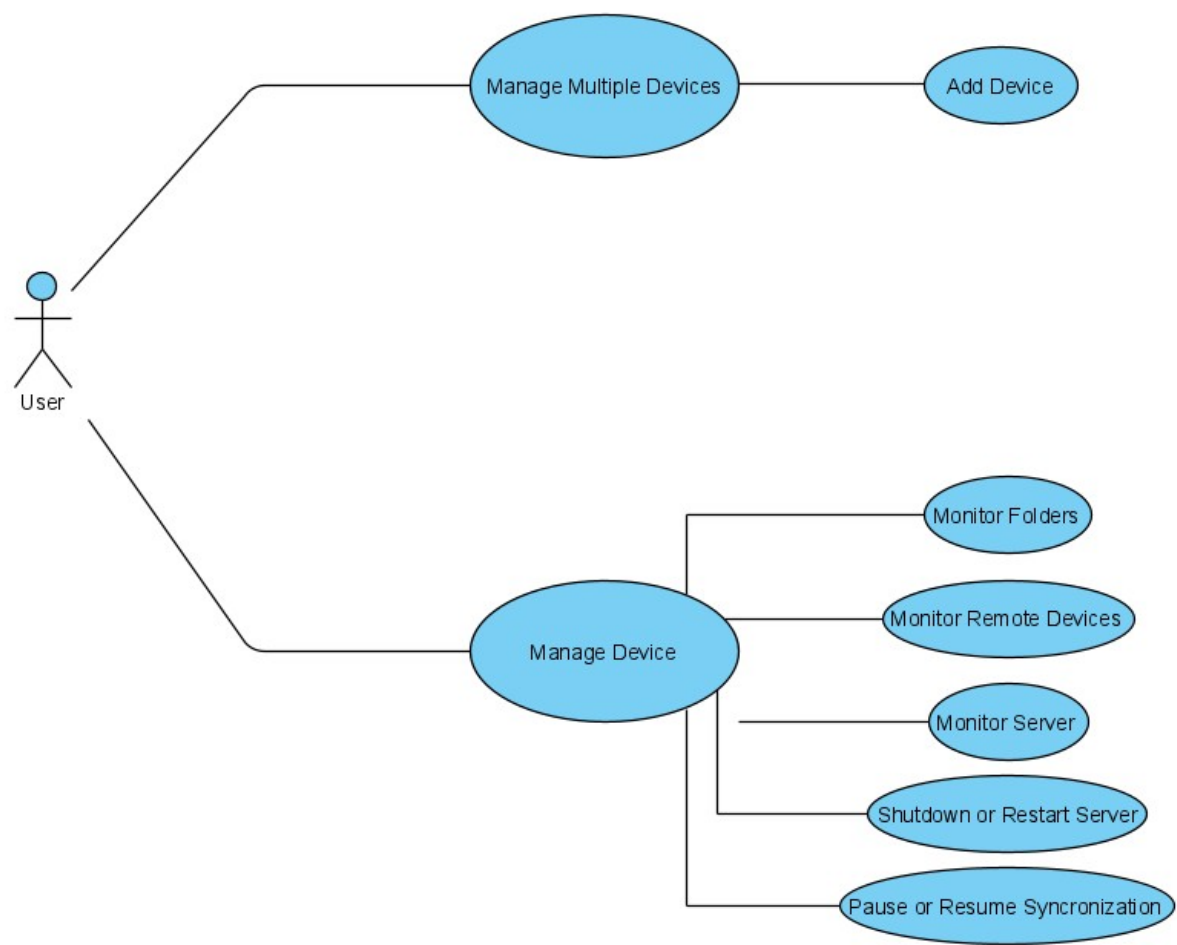
The contribution of the author consists of combining features of the local Syncthing Web configuration interface, Syncthing GTK and other features into an application that allows the management of multiple Syncthing servers from one device, on the go.

---

2 <https://github.com/kozec/syncthing-gtk>

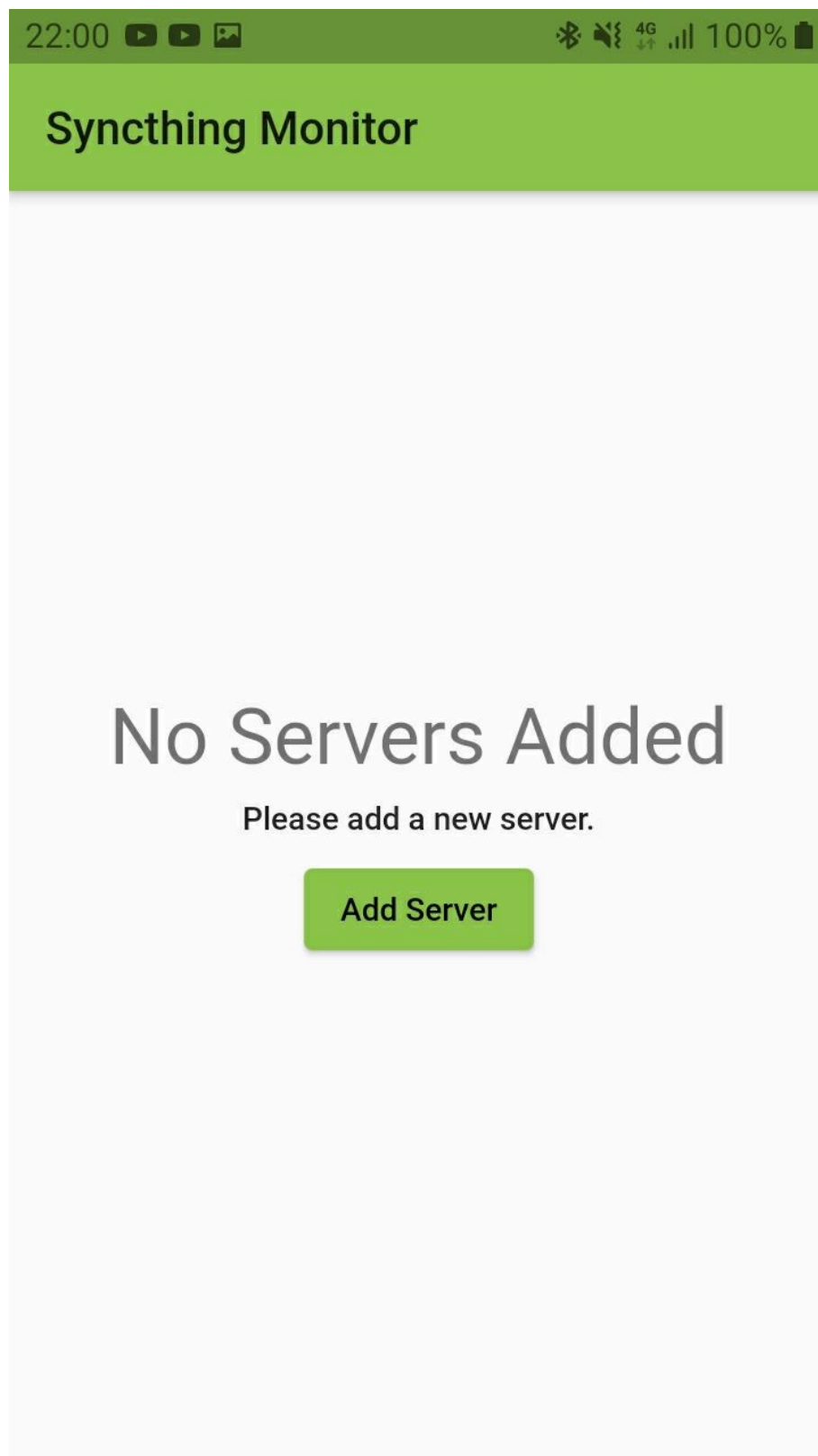
3 <https://play.google.com/store/apps/details?id=com.nutomic.syncthingandroid>

# Functionality



# User Interfaces

## Initial Screen



## Adding a server

21:59 100%

← Add Syncthing Server ✓

Address

---

Name

---

API Key

---

If the name is left empty, it will be determined from the server.


22:01 100%

← Add Syncthing Server

**DC-VPS-Linode**

Syncthing v1.13.1

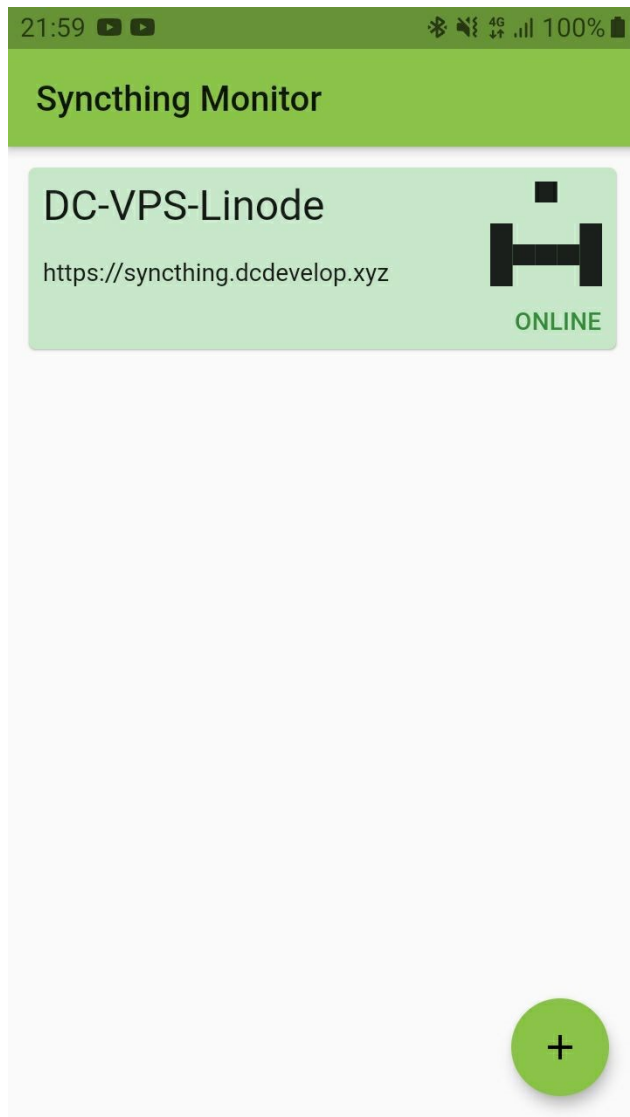
YYTPT7W-OHS6WSZ-CZNBLP5-76GEPBK-USLFEVU-2YQ2MYX-V35ZXKS-K7FGMQ4



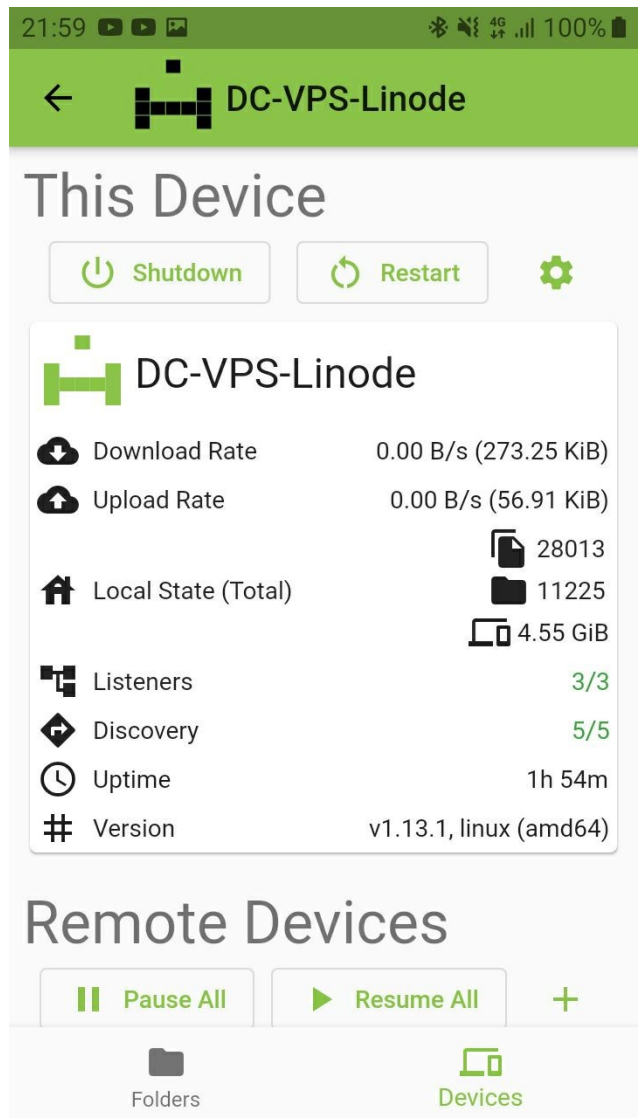
Add



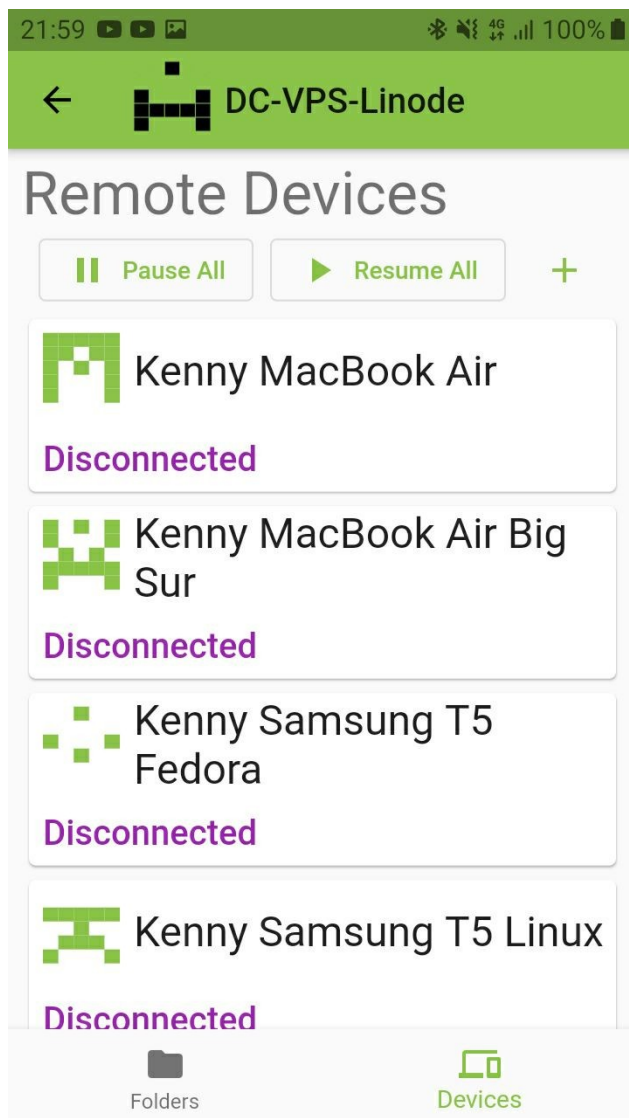
## Main screen with servers added



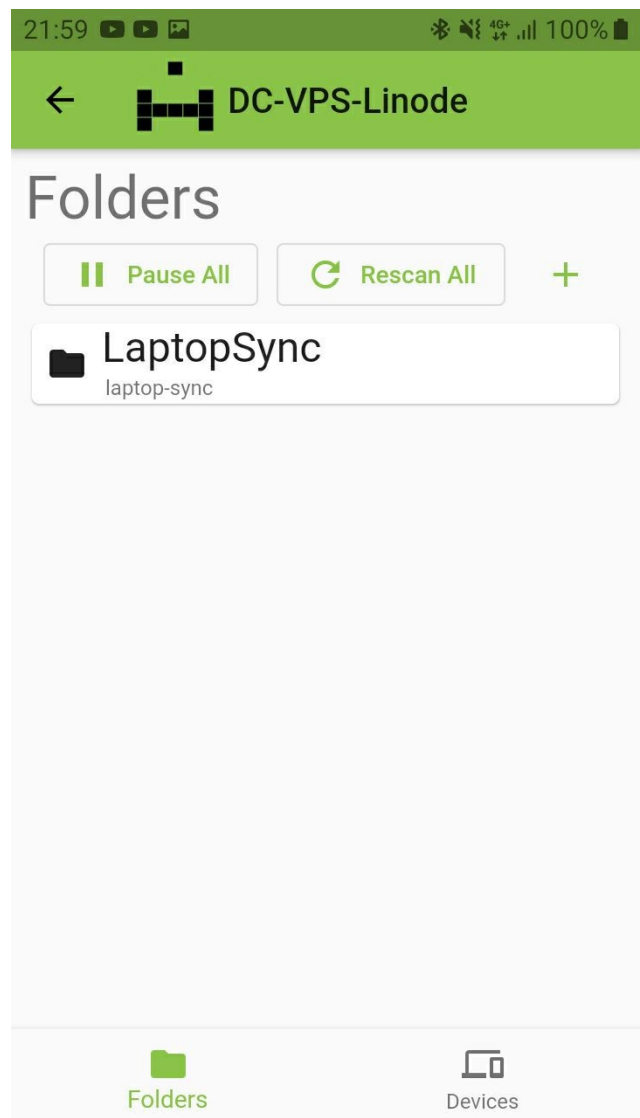
## Monitoring a server



## Monitoring remote devices

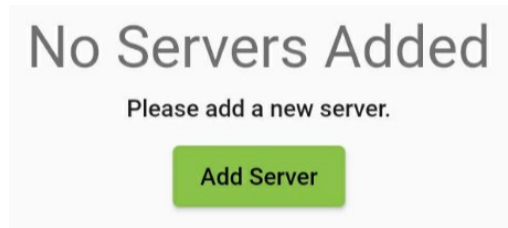


## Monitoring folders

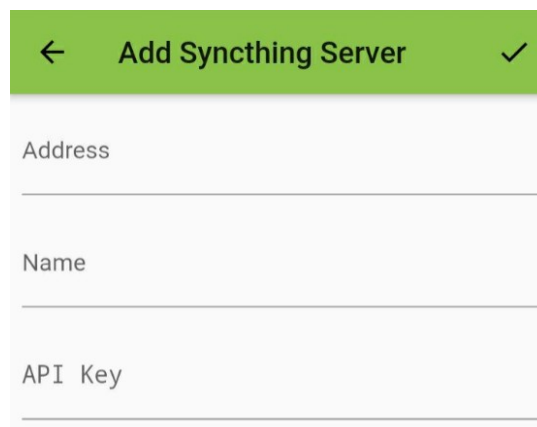


# User's Manual

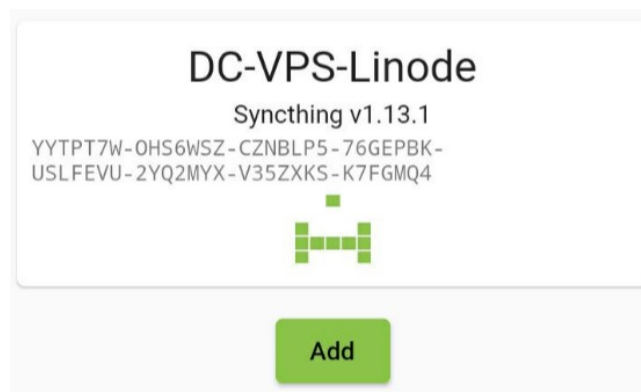
When first opening the app, you will be prompted to add a new server by pressing the button:



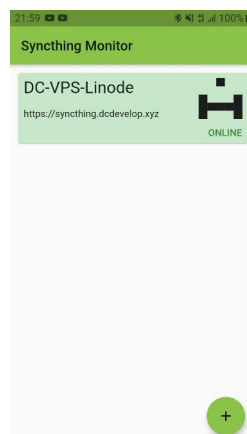
Please fill in the address of the server and the API Key:



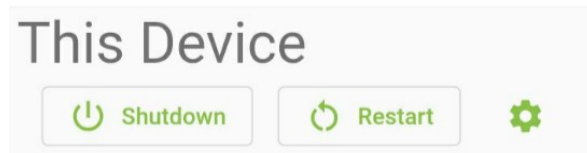
Verify that the server you intend to add is displayed and tap the Add button:



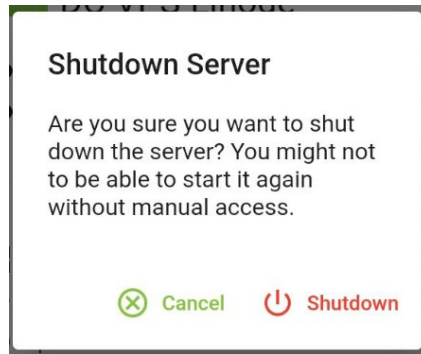
The added device will appear on the main screen. To add an additional server, press the button in the bottom right corner. Tap a server to manage and monitor it.



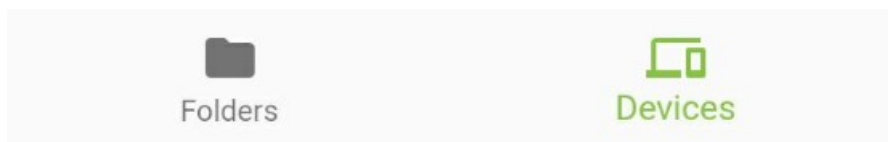
Use the Shutdown or Restart buttons to perform the actions:



If attempting to shut down, you need to confirm the action. Keep in mind that remote access may be unavailable when the server is shut down.



Use the bar at the bottom to switch to the Folders view:



# Technical Manual

The Flutter app contains two main pages in the files “main.dart” and “server\_page.dart”.

The ServersPage class in the “main.dart” file builds the structure of the main page, and displays the cards for each device that was added in the local database.

```
class ServersPage extends StatefulWidget {
  static _ServersPageController of(BuildContext context)
    => _ServersPageController.of(context);

  @override
  _ServersPageState createState() => _ServersPageState();
}

class _ServersPageState extends State<ServersPage> {
  addServer(BuildContext context) async {
    final createdNewServer = await Navigator.of(context).pushNamed("/add_server");
    if (createdNewServer != null) {
      refreshServersList();
    }
  }

  onServerSelection(BuildContext context, SyncthingServerEntry server) async { ... }

  List<SyncthingServerEntry> servers;
  StreamSubscription<List<SyncthingServerEntry>> serversStreamSubscription;
  Timer periodicRefreshTimer;

  Future refreshServersList() async { ... }

  onDebugTools(BuildContext context) async { ... }

  listenToChangeStream(Stream<List<SyncthingServerEntry>> serversChangeStream) { ... }
}
```

For each entry in the database, a tappable card is displayed, represented by the ServerCard class:

```
class ServerCard extends StatelessWidget {
  final SyncthingServerEntry server;
  final Function onServerSelection;

  ServerCard({@required this.server, @required this.onServerSelection});

  Future<_ServerStatus> getServerStatus() async { ... }

  Future<String> getServerName(_ServerStatus status) async { ... }

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: FutureBuilder<_ServerStatus>(
        future: getServerStatus(),
        initialData: _ServerStatus.LOADING,
        builder: (context, statusSnapshot) {
          if (!statusSnapshot.hasData || statusSnapshot.data == _ServerStatus.LOADING) {
            return Card(
              child: Padding(
                padding: const EdgeInsets.all(8.0),
                child: Center(
                  child: CircularProgressIndicator(),
                ), // Center
              ), // Padding
            ); // Card
          }
        },
      ),
    );
  }
}
```

Inside the “server\_page.dart” file, there is one main class that manages the page, and 3 subclasses which manage each section: “This Device”, “Remote Devices” and “Folders”.

“syncthing\_computer\_avatar.dart”, “syncthing\_list\_tile.dart” and “syncthing\_detail\_card.dart” provide building blocks for the common user interface: the computer avatar is the 5x5 logo that identifies each device, the SyncthingListTile is a widget that displays one line of information, and the SyncthingDetailCard is a widget that displays multiple SyncthingListTiles.

## Conclusions and Future Work

While the app is not yet complete, it is already in a minimum usable state.

As described in the **Goal** section, there are still things to be done for the app to be considered complete: ability to change settings of one or multiple servers and ability to configure the files that are ignored from a folder.

## References

- <https://syncthing.net/>
- <https://docs.syncthing.net/dev/rest.html>
- <https://flutter.dev/>
- <https://pub.dev/packages/hive>
- <https://pub.dev/packages/http>
- [https://pub.dev/packages/path\\_provider](https://pub.dev/packages/path_provider)
- <https://pub.dev/packages/path>
- [https://pub.dev/packages/sliver\\_tools](https://pub.dev/packages/sliver_tools)