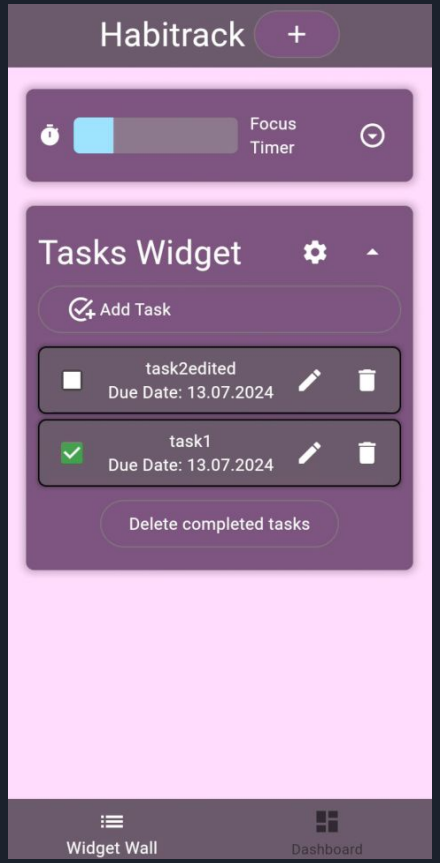
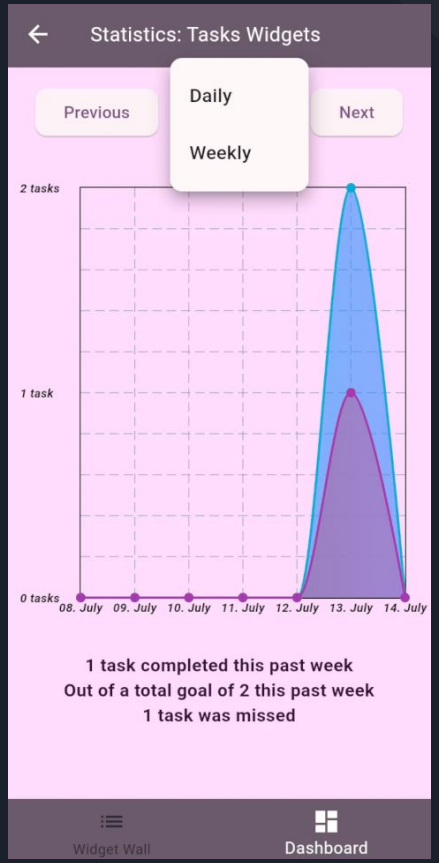
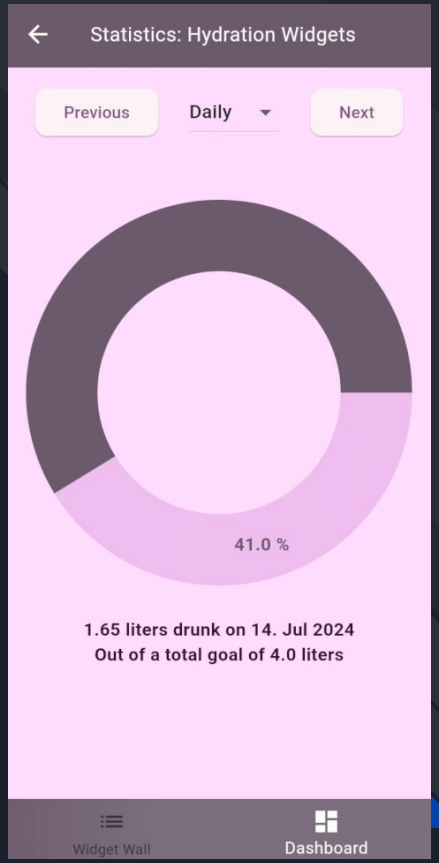
Habitrack

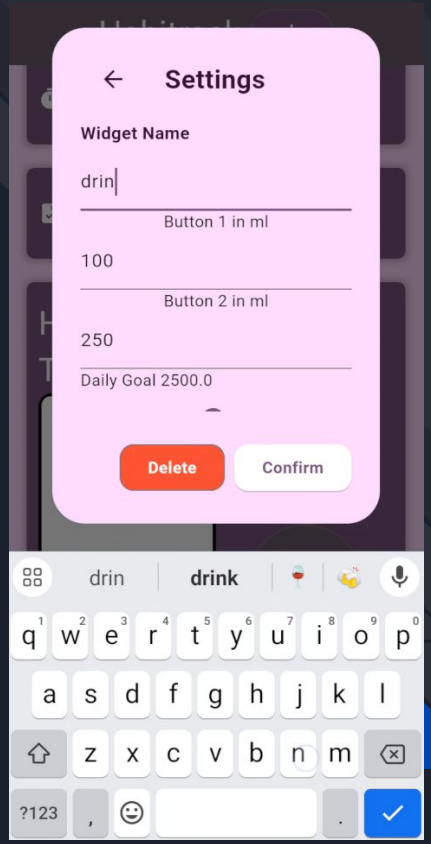

Habitrack
+
Focus Timer
Tasks Widget
Add Task
task2edited
Due Date: 13.07.2024
task1
Due Date: 13.07.2024
Delete completed tasks
Widget Wall
Dashboard


Statistics: Tasks Widgets
Previous
Daily
Weekly
Next
2 tasks
1 task
0 tasks
08. July
09. July
10. July
11. July
12. July
13. July
14. July
1 task completed this past week
Out of a total goal of 2 this past week
1 task was missed
Widget Wall
Dashboard


Statistics: Hydration Widgets
Previous
Daily
Next
41.0 %
1.65 liters drunk on 14. Jul 2024
Out of a total goal of 4.0 liters
Widget Wall
Dashboard

Did&Done

- User Research Interview: Mitschreiber
- Figma design
- Riverpod mit Sembast DB
- Widget funktionalität
- Graph funktionalität



Was war spannend?

- refactoring von refactoring (war überkompliziert)
- zeitdruck: keine Code Freeze gemacht
- ein paar merge conflicts
- gute Arbeit auf individuelle Widgets, aber integration war schwierig

Was war überraschend?

- auch lokales Datenbanken muss asynchron aufgerufen werden, sonst führt es zu Fehlern
- dependency konflikten (flutter_gen vs localization) für freezed
- bei sembast konnte ich nicht composition zum laufen bringen (für die subtasks), musste einfaches String verwenden
- hat nicht auf iOS Safari gelaufen, hatte keine Mitteln es zu debuggen

```
var newItem = searchTerm;
final toConvert = newItem.split(seperator);
final todo = toConvert.elementAtOrNull(0)!;
final due = DateTime.parse(toConvert.elementAtOrNull(1)!);
final createdOn = DateTime.parse(toConvert.elementAtOrNull(2)!);
final completedOn = DateTime.parse(toConvert.elementAtOrNull(3)!);

final completed =
    toConvert.elementAtOrNull(4)!.toLowerCase() == 'true';

final dateFormat = DateFormat('yyyy-MM-dd');
final formattedDate = dateFormat.format(due);
newItem = [
  todo,
  formattedDate.substring(0, 10),
  createdOn,
  completedOn,
  completed,
  false,
].join(seperator);

completedTaskList[i] = newItem;
final controller = ref.watch(homeControllerProvider);
item = item.copyWith(completedTaskList: completedTaskList);
controller.edit(item);
```

```
@override
Widget build(BuildContext context) {
  // this.buildList();
  final controller = ref.watch(itemsProvider);
  switch (controller) {
    case AsyncError(:final error):
      return Text('Error: $error');
    case AsyncData(:final value):
      final allItems = value;
      final items = <dynamic>[];
      for (var i = 0; i < allItems.length; i++) {
        // ignore: avoid_dynamic_calls
```

Was war besonders?

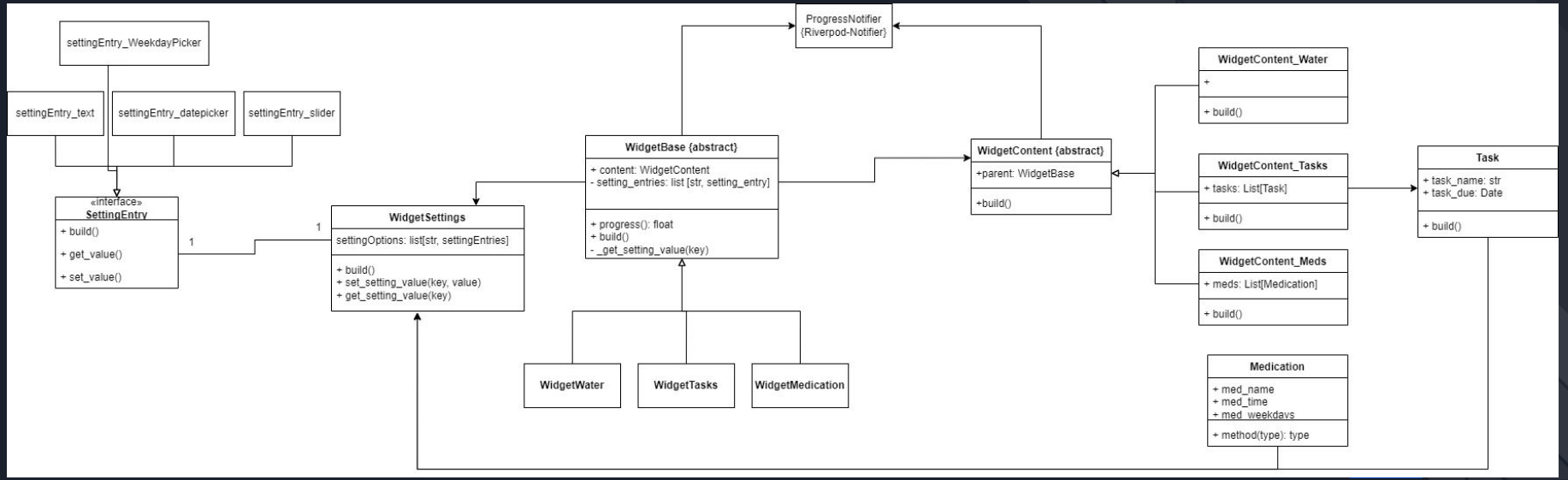
- Trennung zwischen State und Widgets war neu für mich
- habe schon erfahrung mit kleine webapps, wo die DOM manuell aktualisiert müssen
- “write once” und compile für zwei platforms, habe schon PWAs gemacht und von dinge wie react native gehört aber habe niemals etwas wie flutter verwendet: ist sehr praktisch, braucht nur ein paar zusätzliche Linien von Code für platformspezifische Dinge
- gutes platform, “when you get the hang of it” - sehr schönes IDE/Dokumentation integration
- theming mit eine einzige farbe als base viel besser als manuell die Farbe zu setzen
- sembast war straightforward (habe die hälfte der Code von einem Tutorial artikel kopiert)

```
16
17  if (kIsWeb) {
18    final factory = databaseFactoryWeb;
19    final db = await factory.openDatabase('test');
20    runApp(
21      ProviderScope(
22        overrides: [databaseProvider.overrideWithValue(db)],
23        child: const MainApp(),
24      ),
25    );
26    // running on the web!
27  } else {
28    final appPath = await getApplicationDocumentsDirectory();
```

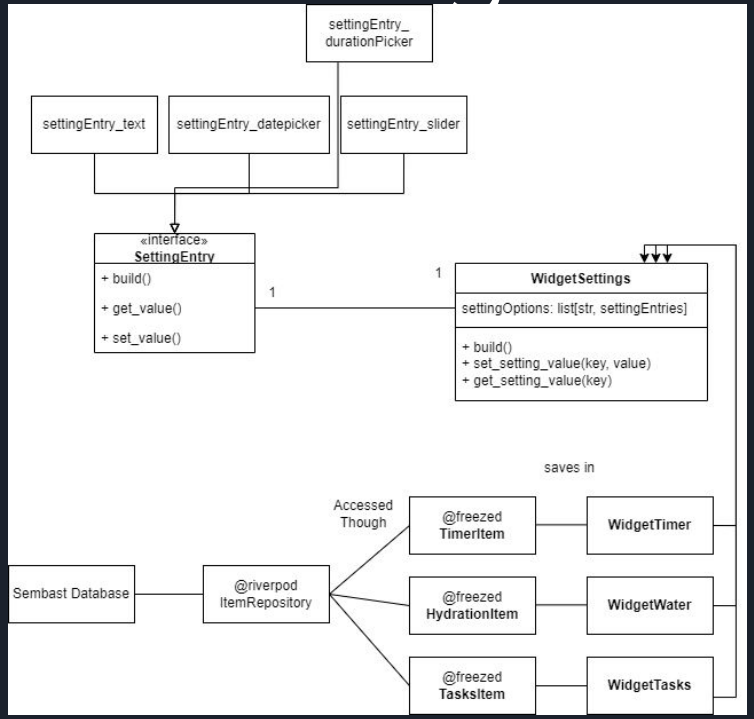
Spannend: Einlernen in Flutter

- Probleme mit dem Initialen Einlernen
- late und final- variablen gescheit benutzen
- von “Python-Stil” wegkommen
- klassendesign: Composition over Inheritance

Implementation: UML-Diagram



Implementation: UML-Diagram 2



Sonstiges

Did&Done:

Interviews: Rednerin

Programmierung:
Routing mit go-router
Design of Software Infrastructure
Theming and Localization
Fixing of VeryGoodAnalysis Problems
Various Touch-Ups

Documentation:
Design
Implementation
Usability Evaluation

Conclusion

- KISS principle (Keep it Simple Silly)
- “failing to plan is planning to fail”