Übung 5 - Protokoll

13.02.22

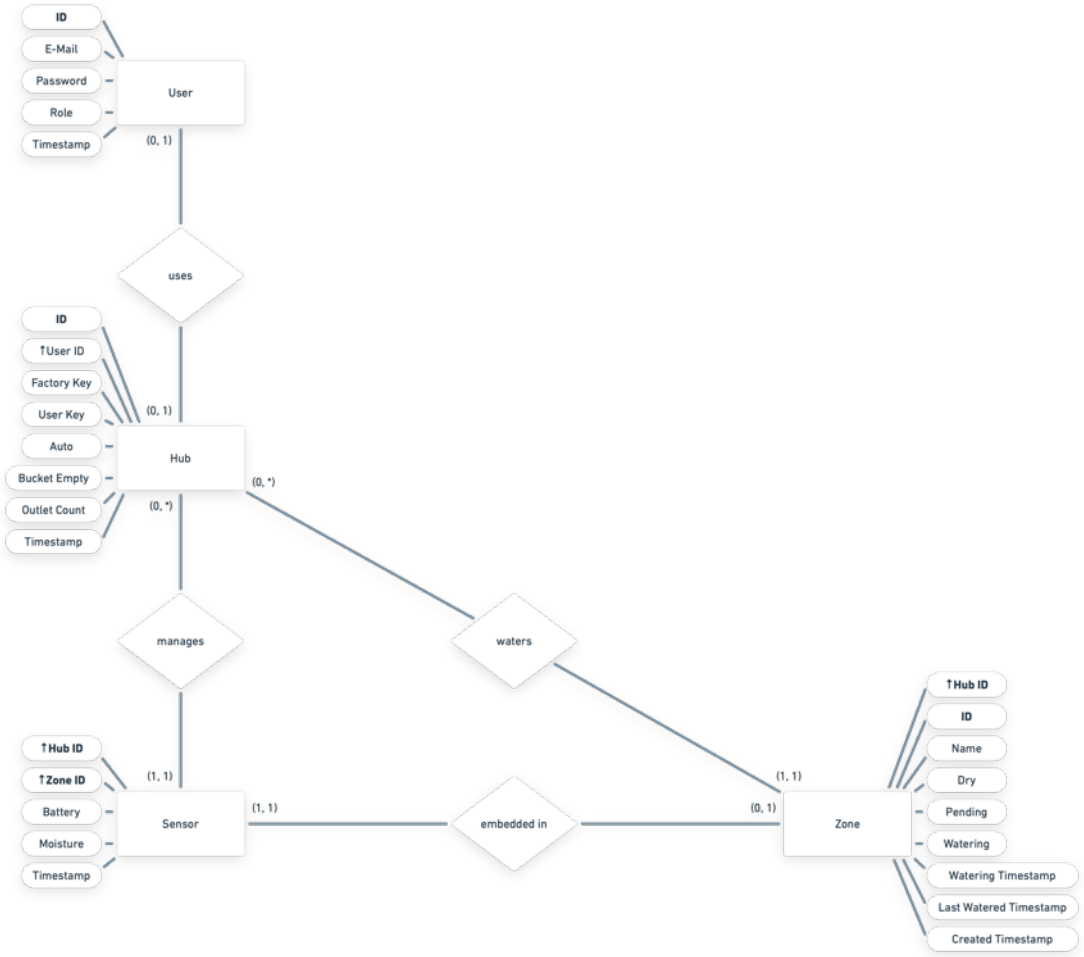
Gruppenname: AppleJuice
Gruppenmitglieder: Albert Kaminski (896587), Dominik Domonell (897035), Sahiram Ravikumar (899517)

Aufgabe 1

Projektname: Bloom

Projektvorschlag:
Wir haben letzten Sommer zusammen an einem Projekt gearbeitet, bei dem wir über SSH und einem Raspberry Pi (Hub), der mit einer Wasserpumpe und Schläuchen verbunden ist, Pflanzen aus der Ferne gießen konnten. Um diese Idee noch benutzerfreundlicher und effizienter zu gestalten, haben wir uns dazu entschlossen, diesen Prozess mithilfe einer App zu vereinfachen. Dazu nutzen wir einen selbsterstellten Webserver, der zwischen dem Raspberry Pi (Hub) und der App geschaltet ist, sodass er zur Kommunikation beider Stacks dient.

ER-Modell des Webservers:



Funktionale Anforderungen:

A.1 Registrieren eines Nutzers

Anforderung: „Als Nutzer möchte ich mit meiner E-Mail Adresse und einem selbst erstelltem Passwort ein Kundenkonto erstellen“
Akzeptanzkriterien: - Der Nutzer muss zur Bestätigung sein Passwort zweimal eingeben und beide Passwörter müssen übereinstimmen - Das Passwort muss aus mindestens 8 Buchstaben und einem Sonderzeichen bestehen

A.2 Login eines Nutzers

Anforderung: „Als Nutzer möchte ich mich mit meiner E-Mail Adresse und meinem selbst erstelltem Passwort anmelden“
Akzeptanzkriterien: - Die E-Mail Adresse mit dem Passwort müssen vorher über den Registrierungsprozess in der Datenbank sein - Die Daten müssen mit den Daten aus der Datenbank übereinstimmen, ansonsten erfolgt eine Fehlermeldung

A.3 Hinzufügen eines Hubs zum Nutzer

Hardcoded: Das Hinzufügen und die Verknüpfung mit dem Benutzer eines Hubs erfolgt aus Einfachheit über den Webserver und nicht über die App

Anforderung: „Der Nutzer möchte einen Hub (Raspberry Pi) erstellen und mit seinem Benutzerkonto verbinden“
Akzeptanzkriterien: - Der Nutzer muss vorher in der Datenbank vorhanden sein, sodass die Verbindung mit dem Hub möglich ist.

A.4 Hinzufügen und Entfernen von Zonen und den damit verbundenen Sensoren zum Hub

Hardcoded: Das Hinzufügen einer Zone erfolgt aus Einfachheit über den Webserver und nicht über die App

Anforderung: „Als Nutzer möchte ich Zonen hinzufügen (die ihren eigenen Sensor besitzen), sodass ich diese einzeln ansteuern kann beim Gießen“

Akzeptanzkriterien:
- Die Zone muss mit dem Hub über das Attribut hub_id verbunden sein, sodass eine eindeutige Zuordnung vorliegt
- Die Zone muss eine eigene ID besitzen, damit die Verbindung zum Sensor erfolgt
- Zur Benutzerfreundlichkeit besitzt jede Zone ihren eigenen Namen
- Eine Zone hat die Attribute, ob es zu trocken ist (Boolean), ob gegossen wird (Boolean) und wann zuletzt gegossen wurde (Date)

A.5 Auflisten aller Zonen eines Hubs

Anforderung:
„Als Nutzer möchte ich alle Zonen sehen, die zu meinem Hub hinzugefügt wurden“

Akzeptanzkritierien:
- Als Übersicht werden Zonenname und ID angezeigt
- Durch Klicken auf die Zone erfolgt eine detaillierte Ansicht der Zone, sodass man sie einzeln ansteuern kann

A.6 Gieß -und Stoppfunktion

Anforderung:
„Als Nutzer möchte ich eine Zone gießen und das Gießen anhalten, falls notwendig“

Akzeptanzkriterien:
- Das Gießen und Stoppen erfolgt über einen Button

A.7 Anzeige des derzeitigen Feuchtigkeitswert und Gießstatus einer Zone

Anforderung:
„Als Nutzer möchte ich den aktuellen Feuchtigkeitsstatus meiner Zone sehen, sowie den Status, ob gerade gegossen wird und wann zuletzt gegossen wurde“

Akzeptanzkriterien:
- Die Anzeige des Feuchtigkeitswerts wird in Prozent dargestellt
- Der Zeitpunkt, wann die Messung des Feuchtigkeitswerts war, wird als Text angezeigt
- Ob gerade gegossen wird, wird als Text angezeigt
- Ein Timer ist eingebaut, damit das Gießen automatisch nach einer Zeit gestoppt wird
- Der Zeitpunkt wann zuletzt gegossen wurde, wird als Text angezeigt

A.8 Anzeige, ob die Batterie der Sensoren leer ist

Funktion zurzeit nicht notwendig, da die Batterien der Sensoren neun Monate halten

Anforderung:
„Als Nutzer möchte ich sehen, ob die Batterien meiner Sensoren leer sind, sodass ich diese wenn nötig wechseln kann“

Akzeptanzkriterien:
- Falls die Batterien leer sind, wird ein rotes Batteriezeichen mit einem kleinen Warntext angezeigt

A.9 Benachrichtigung, wenn der Wasserstand des Eimers vom Hub zu niedrig ist

Anforderung: „Als Nutzer möchte ich benachrichtigt werden, falls mein Wassereimer einen niedrigen Wasserstand erreicht“
Akzeptanzkriterien: - Eine Meldung wird in der Zonenübersicht angezeigt (Dashboard)

A.10 Benachrichtigung, falls eine Zone zu trocken ist

Anforderung: „Als Nutzer möchte ich benachrichtigt werden, falls eine meiner Zonen einen niedrigen Feuchtigkeitswert erreicht“
Akzeptanzkriterien: - Eine Meldung wird in der Zonenübersicht angezeigt (Dashboard)

A.11 Option zur automatischen Bewässerung aller Zonen

Anforderung: „Als Nutzer möchte ich die Möglichkeit besitzen, meine Pflanzen automatisch, anhand der Feuchtigkeitswerte gießen zu lassen“
Akzeptanzkriterien: - In der Zonenübersicht (Dashboard) gibt es die Möglichkeit die Funktion ein -und auszuschalten (Attribut im Hub) - Die Funktionalität für das automatisierte Gießen liegt im Webserver

A.12 Ändern des Namens einer Gießzone

Funktion zurzeit nicht notwendig

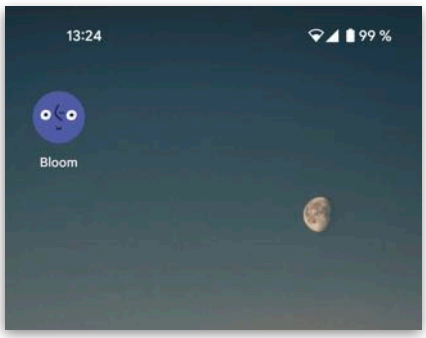
Anforderung: „Als Nutzer möchte ich die Möglichkeit besitzen, den Namen meiner Zonen, wenn nötig zu ändern“
Akzeptanzkriterien: - In der Detailansicht einer Zone kann über ein Icon der Namen geändert werden

A.13 Responsives Design, sodass die App auf allen Geräten möglich ist

Anforderung: „Als Nutzer möchte ich die Möglichkeit besitzen, die App auf jedem Android Gerät nutzen zu können“
Akzeptanzkriterien: - Die App muss responsiv gestaltet sein, um die Funktion ermöglichen zu können

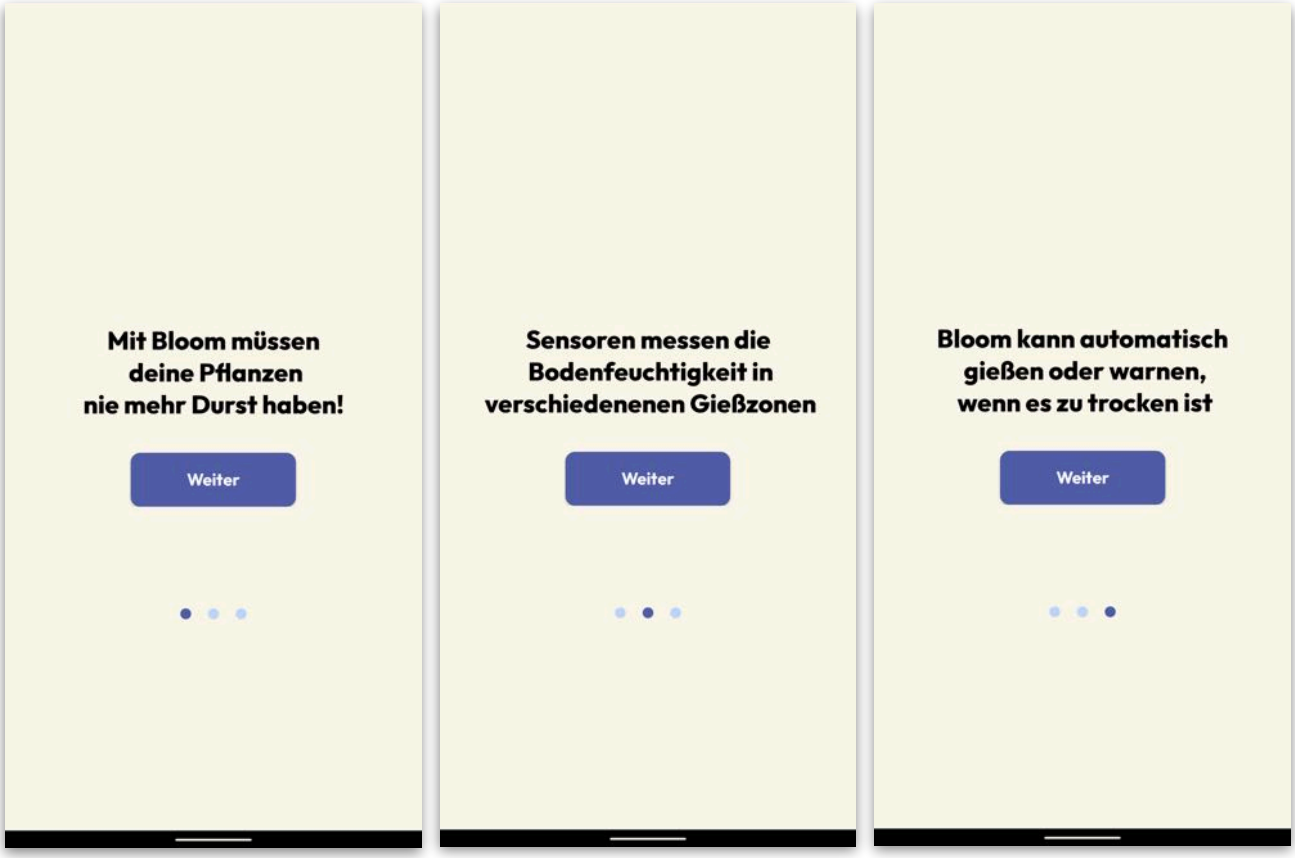
Aufgabe 2

Screenshots der Oberflächen & Beschreibung des Bedienungsablaufs:



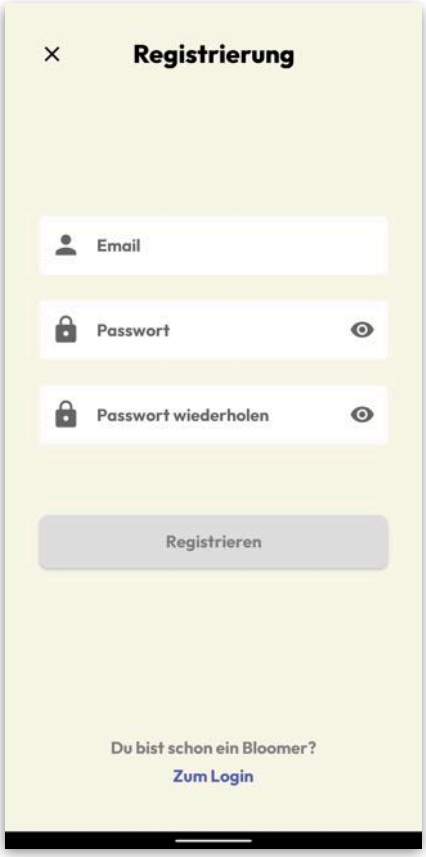
1. Durch Klicken des selbst gestalteten App-Icons gelangt man zur App.

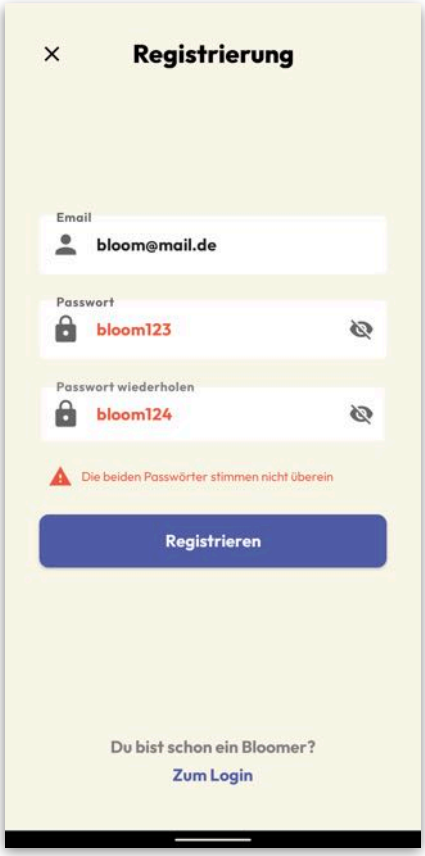
2. Zuerst öffnet sich der Splash-Screen mit einer Animation.

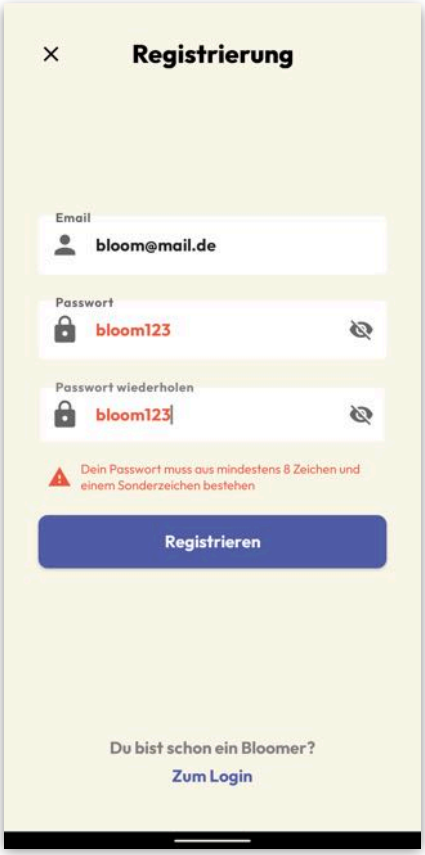


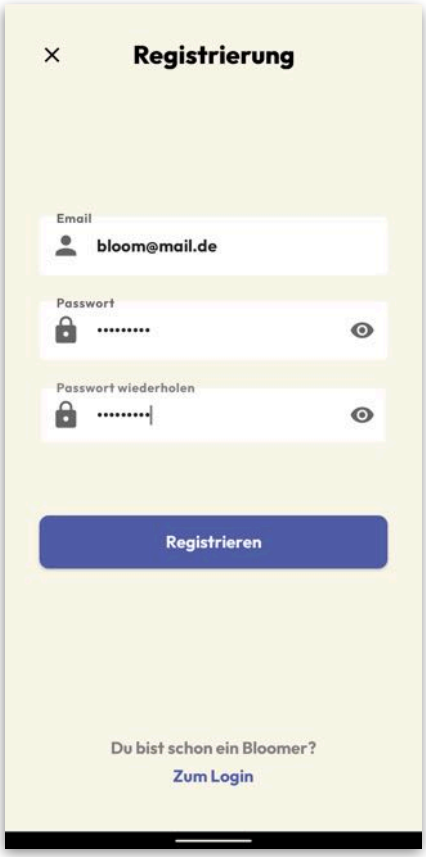
3. Man wird zu den OnBoarding-Screens weitergeleitet, in welchen kurz und benutzerfreundlich beschrieben wird, was unser App Bloom ist und welche Kernfunktion sie besitzt. Durch den „Weiter“-Button gelangt man dann zur nächsten Seite.

5. Auf der Willkommensseite hat man dann die Möglichkeit über den Registrierungslink zu der Registrierungsseite zu kommen oder über den Login-Button zur Loginseite zu kommen.

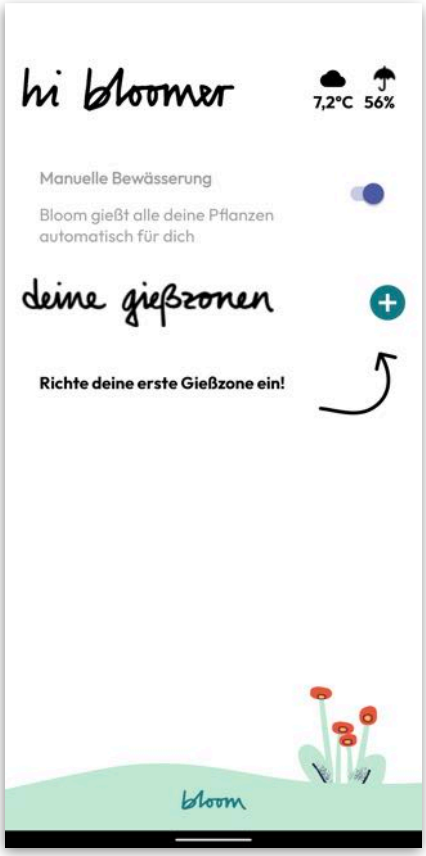




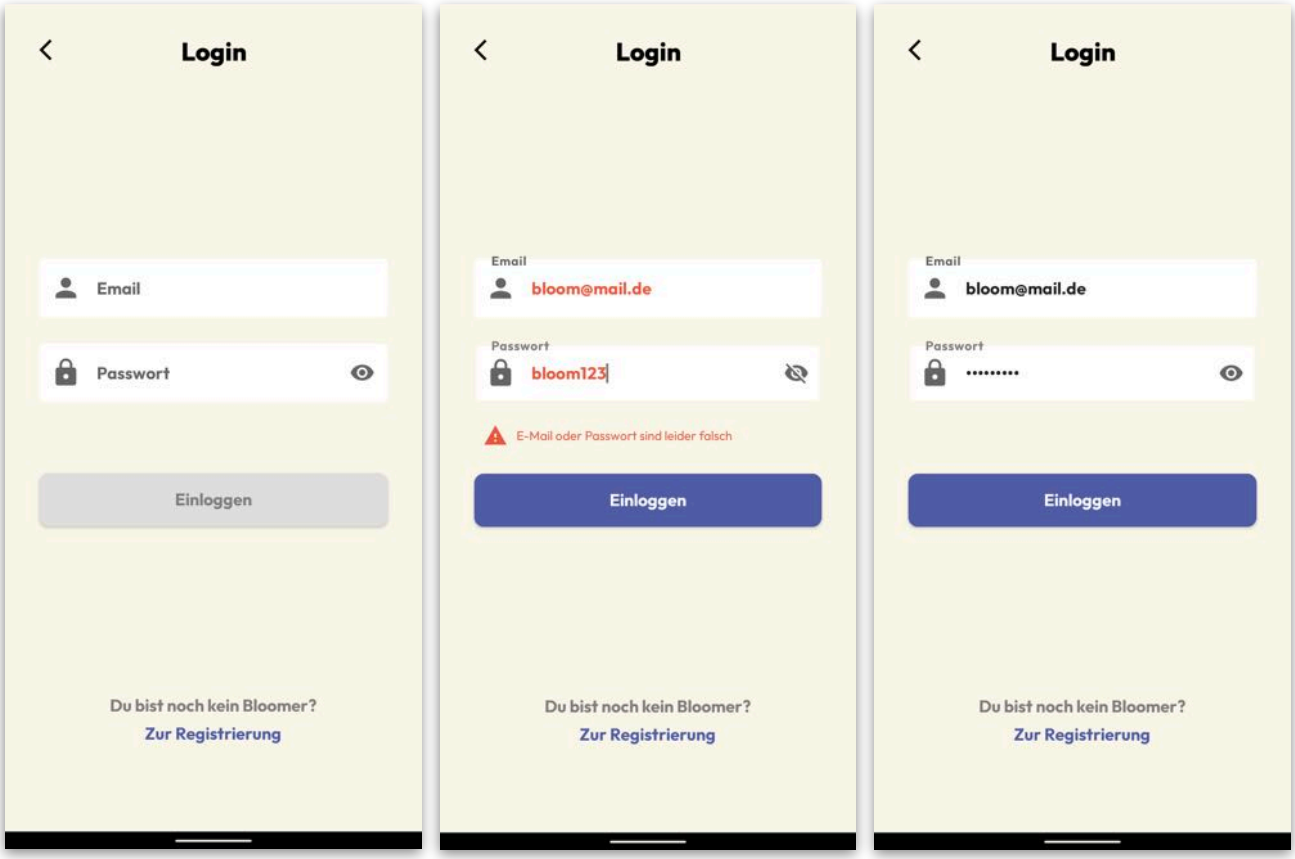




6. Im Registrierungsscreen muss der Nutzer seine E-Mail und sein Passwort zweimal eingeben. Erst wenn er das letzte Eingabefeld ausgefüllt hat, ändert sich die Buttonfarbe von grau zu blau. Falls die Passwörter nicht übereinstimmen wird eine Fehlermeldung ausgegeben. Falls das Passwort keine 8 Zeichen lang ist und keine Sonderzeichen enthält wird auch eine Fehlermeldung ausgegeben.
(A.1 Registrieren eines Nutzers)



7. Nachdem der Nutzer sich registriert hat, wird er zum Dashboard weitergeleitet. Zurzeit ist noch keine Zone eingerichtet. Die Zone und die dazugehörigen Sensoren haben wir über den Server hinzugefügt.

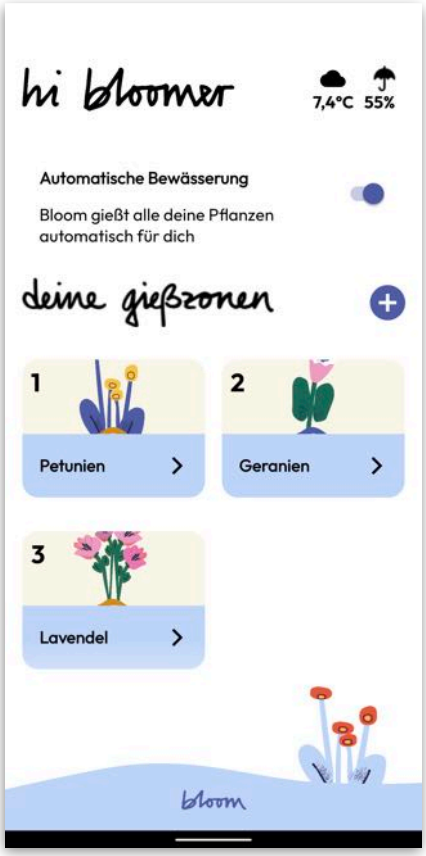


8. Auf der Loginseite kann der Nutzer seine Daten eingeben, falls diese nicht mit den Daten vom Server übereinstimmen, wird eine Fehlermeldung ausgegeben.
(A.2 Login eines Nutzers)

9. Auf dem Dashboard kann der Nutzer seine ganzen eingerichteten Zonen seines Hubs sehen. **(A.5 Auflisten aller Zonen eines Hubs)**
Man bekommt das aktuelle Wetter zu sehen (Temperatur, Regenwahrscheinlichkeit. Eine Benachrichtigung wird angezeigt, weil eine Zone zu trocken ist. **(A.10 Benachrichtigung, falls eine Zone zu trocken ist)**

10. Sobald der Nutzer auf eine Zone geklickt hat, erscheint eine detaillierte Ansicht dieser Zone.
(A.7 Anzeige des derzeitigen Feuchtigkeitswert und Gießstatus einer Zone)

11. Wenn der Nutzer auf den Gieß-Button drückt, startet der Gießvorgang auf dem Server und der Timer, der synchron zum Server ist, zeigt die verbleibende Zeit an, bis der Gießvorgang automatisch beendet wird. Der Nutzer kann diesen Vorgang auch selber über den Stop-Button beenden.
(A.6 Gieß -und Stoppfunktion)



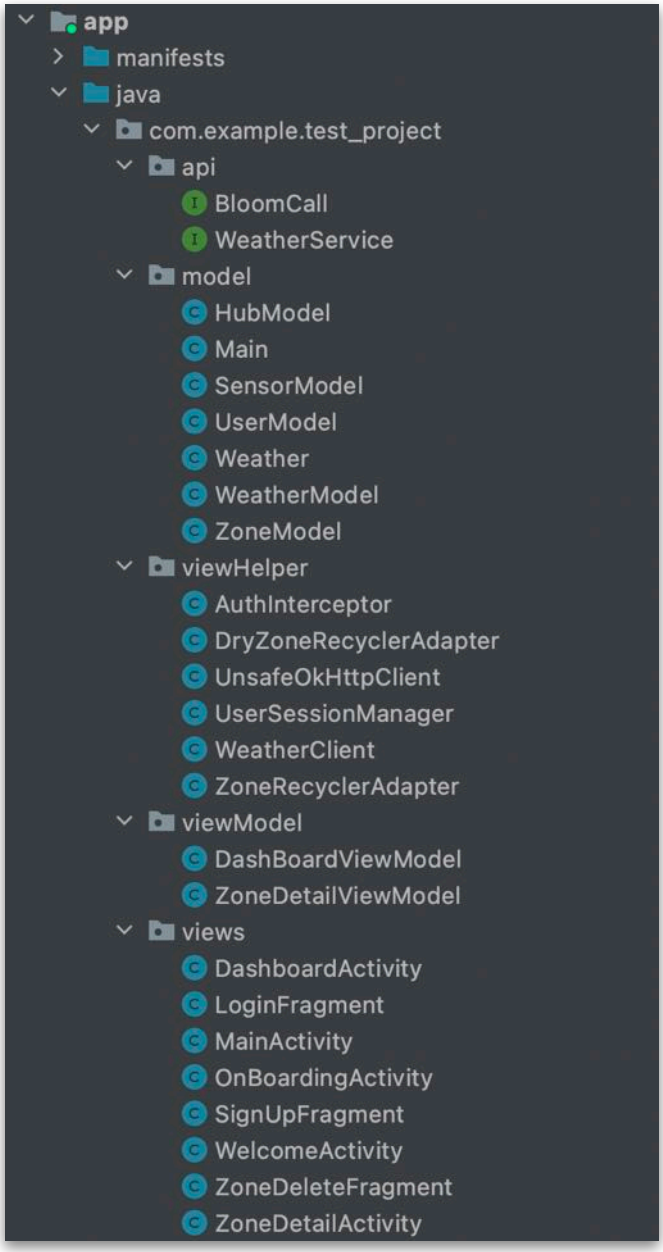
12. Zurück auf dem Dashboard kann der Nutzer über einen Switch die manuelle Bewässerung auf automatische Bewässerung stellen. Sobald ein niedriger Feuchtigkeitswert erreicht wird, wird der Gießvorgang auf dem Server gestartet.
(A.11 Option zur automatischen Bewässerung aller Zonen)

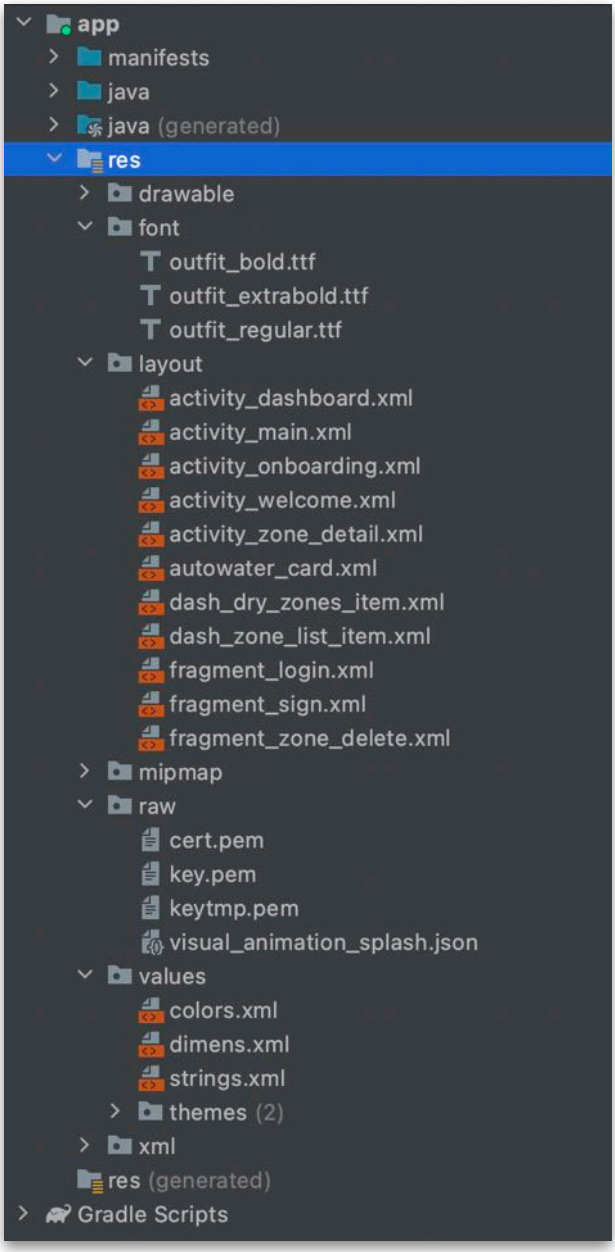
13. Die Detailansicht im automatischen Modus sieht dann folgendermaßen aus. Für Zone 3 wurde der Gießvorgang gestartet, weil der Feuchtigkeitswert einen niedrigen Stand erreicht hat, auch hier kann dieser über den Stopp-Button beendet werden.

14. Über den Text „Gießzone löschen“ kommt der Nutzer zur Möglichkeit eine Gießzone zu löschen. Sobald der Button gedrückt wird, wird die Zone mit dem verknüpften Sensor aus der Datenbank gelöscht. **(A.4 Entfernen von Zonen und den damit verbundenen Sensoren zum Hub)**
Die Zone wird dann nicht mehr auf dem Dashboard angezeigt. Außerdem ist eine Meldung zu sehen, dass der Wassertank des Hubs leer ist.
(A.9 Benachrichtigung, wenn der Wasserstand des Eimers vom Hub zu niedrig ist)

A.13 Responsives Design, sodass die App auf allen Geräten möglich ist
Beispielhaft für das Pixel 2 (411 x 731)

Ordnerstruktur:





Ausführliche Dokumentation der Attribute und Methoden befinden sich als Kommentare in den Dateien.

Arbeitsaufteilung:

Albert Kaminski:
- Anbindung der App an unseren selbsterstellten Server:
  - Interface, welches die HTTP Requests für den Server enthält *(BloomCall.java)*
  - Implementierung der vom Server erforderlichen Model Klassen in GSON *(HubModel.java, SensorModel.java, UserModel.java, ZoneModel.java)*
  - Einbau eines HTTP Clients, sodass https Anfragen verarbeitet werden *(UnsafeOkHttpClient.java)*
  - Einbau eines Interceptor, sodass Tokenhandling über App möglich ist *(AuthInterceptor.java)*
  - Einbau eines SessionManagers, sodass eine Benutzersession möglich ist und zur Speicherung von Daten (Token) in den SharedPreferences *(UserSessionManager.java)*
- Anbindung einer Wetter API, sodass über den Standort die Wetterdaten auf dem Dashboard angezeigt werden *(WeatherService.java, WeatherClient.java)*:
  - Implementierung der dafür erforderlichen Model Klassen in GSON *(Main.java, Weather.java, WeatherModel.java)*

Dominik Domonell:
- Erstellung des Designs über Figma-Prototypen und einem Styleguide:
  - Erstellung der Grafiken (Illustrationen, Icons) in *drawable*
  - Einbindung einer eigenen Schrift in *font*
  - Implementierung des Designs, der Farben, der Texte und Maße nach den Design Prototypen *(colors.xml, dimens.xml, strings.xml, themes.xml)*
- Implementierung eines Splashscreens mit Animation *(MainActivity.java, activity_main.xml, visual_animation_splash.json)*
- Implementierung einer OnBoardingsseite, die zur Anleitung / Erklärung dient *(OnBoardingActivity.java, activity_onboarding.xml)*
- Implementierung einer Willkommensseite, die zu den anderen Screens weiterleitet *(WelcomeActivity.java, activity_welcome.xml)*
- Implementierung einer Registrierungsseite, mit der man sich über den Server registrieren kann *(SignUpFragment.java, fragment_sign.xml)*
- Implementierung einer Anmeldungsseite, mit der man sich über den Server anmelden kann *(LoginFragment.java, fragment_login.xml)*

Sahiram Ravikumar:
- Erstellung einer Listenansichtsseite, welche dynamisch eine Übersicht der Zone eines Hubs anzeigt: *(DashboardActivity.java, activity_dashboard.xml)*
  - Implementierung einer RecyclerView, welche dynamisch alle Zonen als Kachel darstellt *(ZoneRecyclerAdapter.java, dash_zone_list_item.xml)*
  - Implementierung einer RecylcerView, welche dynamisch alle Zonen anzeigt, die zu trocken sind *(DryZoneRecyclerAdapter.java, dash_dry_zones_item.xml)*
  - Implementierung einer Switch Funktion, welche den Auto Modus ein -und ausschaltet (autowater_card.xml)
  - Implementierung einer Warnnachricht, welche angezeigt wird, wenn der Wassertank leer ist

- Erstellung einer Detailansichtsseite, welche dynamisch eine detaillierte Ansicht einer Zone anzeigt: *(ZoneDetailActivity.java, activity_zone_detail.xml)*
  - Implementierung des Layouts, das je nach Modus (manuell, auto) das Design ändert
  - Anzeige des Namens, der ID, der Feuchtigkeit in Prozent, des Zeitpunkts der letzten Messung
  - Anzeige des Zeitpunkts des letzten Gießens, Einbau eines Timers, der mithilfe eines TimeStamp vom Server errechnet wird
  - Einbau der Gieß -und Stoppfunktion über einen Button
- Erstellung einer Löschseite, welche das Löschen einer Zone (mit dem verbundenen Sensor) auf dem Webserver ermöglicht *(ZoneDeleteFragment.java, fragment_zone_delete.xml)*
- Implementierung von ViewModel Klassen, welche über das Observer-Pattern, die View ändern *(DashboardViewModel.java, ZoneDetailViewModel)*

Dokumentation des erreichten Wissensstandes und der Lernerfolge:

Albert Kaminski:
Vor der Übung konnte ich nur einfache Applikationen umsetzen, doch durch Übung 5 ist es mir gelungen, das Konzept von HTTP-Anfragen von App zu einem Server mehr zu verstehen und diese Kopplung auch durch einen HTTP-Client zu implementieren. Insbesondere habe ich mich um das Token Handling via eines Interceptors gekümmert, sowie der Persistierung dieser Daten durch einen Sessionmanager und der erforderlichen Model-Klassen in GSON. Als letztes habe ich noch gelernt, wie man eine fremde API anbindet und dieses neu erlangte Wissen auch bei einer externen Wetter API umgesetzt, um Temperatur und Regenwahrscheinlichkeit abzurufen. Mein Wissensstand hat sich besonders im Bereich Backend extrem verbessert und mir gezeigt das Apps nicht nur statische Templates abbilden müssen, sondern auch dynamisch Daten empfangen und bearbeiten können.

Dominik Domonell:
In der letzten Übung habe ich mich viel mit dem Design und dessen Implementierung beschäftig und gelernt, wie man ein Design einheitlich und intuitiv in Android Studio umsetzt. Dazu habe ich außerdem gelernt, wie man Views responsiv gestaltet und auch wie man den Workflow beschleunigt, indem man viel mit globalen Variablen arbeitet. Zudem habe ich gelernt, wie man mit Calls arbeitet und so die Registrierung und Anmeldung implementiert. Außerdem habe ich gelernt wie man mit Fragments arbeitet und wie man JSON Animationen in Android Apps einbaut, um somit einen Splashscreen realisieren zu können. Die Erstellung von Wireframes in Figma hat mir sehr geholfen das endgültige Design zu planen und umzusetzen. Dazu habe ich die Designelemente in Illustrator gestaltet und in Android Studio eingefügt. Zudem habe ich im Allgemeinen gelernt, wie man nicht nur Designelemente einfügt und verwendet, sondern auch eigene Schriften.

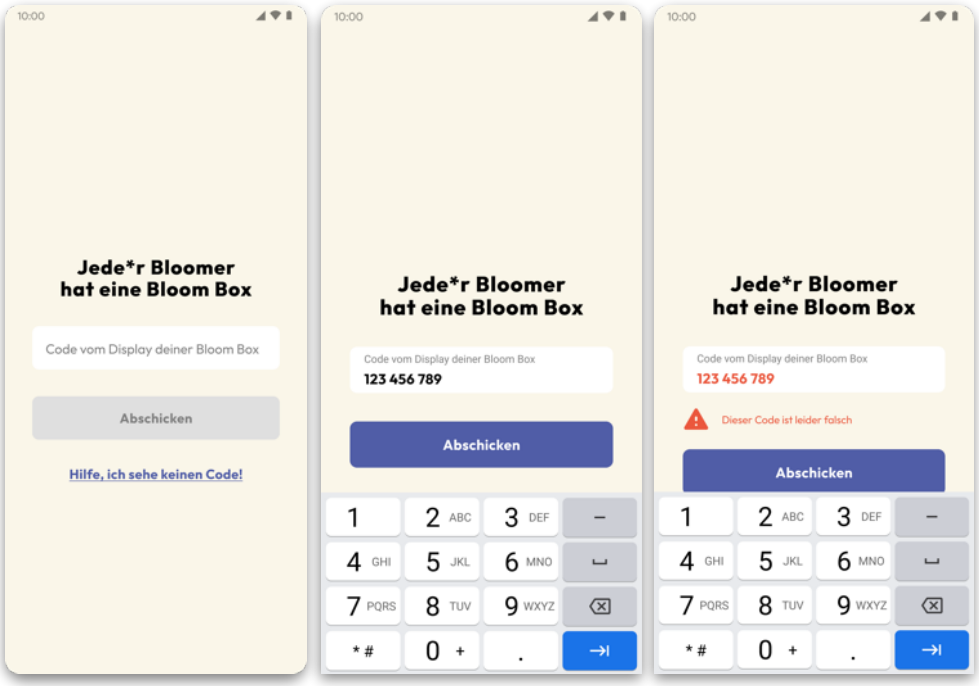
Sahiram Ravikumar:
Vor der Übung wusste ich nur wie man einfache Android Anwendungen programmiert, die nicht viele Funktionalitäten hatten und nur Informationen auf dem Display anzeigen konnten, nicht wirklich responsiv waren und auch nicht zusammenhängende Fragments bzw. aufeinander basierende Seiten hatten. Durch Übung 5 konnte ich mir aneignen, wie

man Daten, die man von einem Server erhält, dynamisch in einer View darstellen kann (auch mithilfe von RecyclerViews), wie man diese Daten auch verarbeiten kann (mithilfe von einem Switch) und wie man Daten zurück zum Server schickt (über einen Gießbutton). Grundlegend musste ich mich dafür mit HTTP-Client, Retrofit und Calls beschäftigen. Zudem konnte ich lernen, wie man Listen-Views erstellt und dazu noch die verknüpften Detail-Views, welche Informationen aus der vorherigen View vererbt bekommen. Des Weiteren konnte ich zum ersten Mal das MVVM-Pattern in einem Projekt umsetzen, um somit wichtige Funktionen (wie GET Anfragen) in die ViewModel-Klassen zu verschieben und auch mit Observer zu arbeiten, die die View nur ändern, wenn sich etwas in den Daten ändert. Außerdem hab ich verstanden, wie man mit (JavaScript und Java) TimeStamps arbeitet und diese nutzen kann, um die Dauer dynamisch anzuzeigen und auch einen Timer zum Laufen zu bringen. In Hinsicht auf Design hab ich gelernt, wie man responsive Apps gestaltet und wie die ganzen Layouts (wie Constraint, Linear -und RelativeLayout) wirklich funktionieren, wie man (Drawables oder Farben von) UI-Elemente über die Views ändert und anpasst. Ich habe verstanden, wie Fragments und Activities zusammenhängend funktionieren und wie man untereinander Daten weitergibt.

Ausblicke (aus der Lehrveranstaltung):

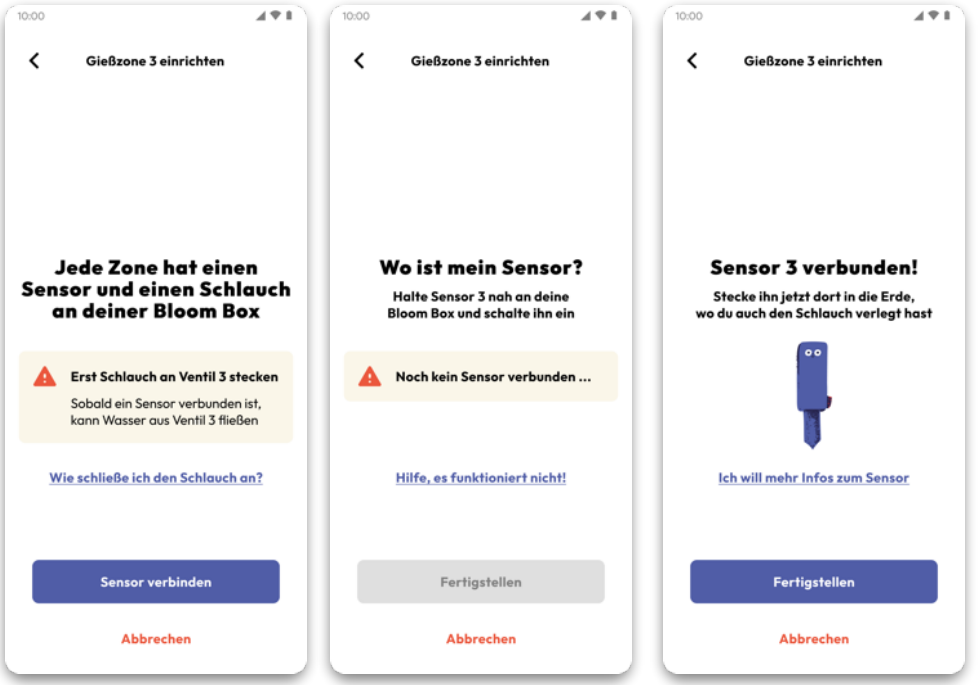
Wir wollen auf jeden Fall in den Semesterferien die App soweit fertig kriegen, dass alle funktionalen Anforderungen umgesetzt sind. Anbei sind Figma Prototypen, wie die Screens aussehen könnten, die wir nicht umgesetzt haben. Wir werden dann auch daran arbeiten den Hub, also den Raspberry Pi, mit Feuchtigkeitssensoren auszustatten und voll funktionstüchtig zu machen, sodass dieser auch Daten an unseren Webserver schicken kann. Zudem wollen wir eine Datenbank mit verschiedenen Pflanzenarten einbinden, die dann Einfluss auf den Gießalgorithmus haben. Zudem wäre es mit Blick auf die Zukunft von Vorteil unseren Raspberry Pi um ein LTE-Modul zu erweitern, damit man die Daten aus dem Garten immer in der App überblicken kann.
Eine weitere Idee wäre es sich mit Push-Benachrichtigungen zu beschäftigen und diese in der App umzusetzen. Außerdem haben wir die Idee die App auch für iOS Geräte zu erstellen, weil wir alle dieses Semester auch das Modul dafür belegt haben.
Des weiteren könnte man als visuelles Feature eine Feuchtigkeitshistorie als Verlaufsdiagramm darstellen, ähnlich wie wir es schon in Übung 3 gemacht haben mit den Standortdaten. Ein Dark Mode wäre wahrscheinlich auch sehr einfach umsetzbar und würde zur heutigen Designsprache passen. Es gibt auch noch die Möglichkeit mittels einer Google Lens Verknüpfung, Pflanzen zu erkennen und diese dann beim Hinzufügen einer Zone (als Zonenname und präferierten Feuchtigkeitwerte) zu benutzen.

Figma Prototypen von den Funktionalitäten, die wir nicht umgesetzt haben:



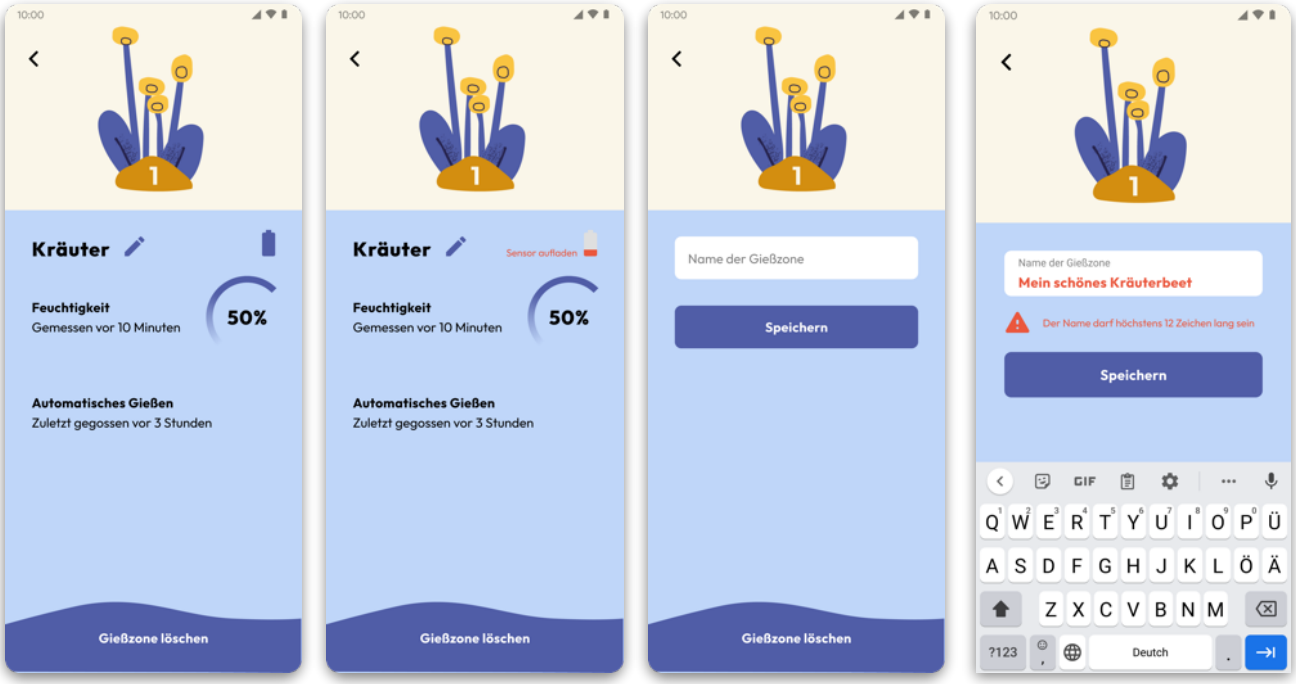
A.3 Hinzufügen eines Hubs zum Nutzer

Über die Eingabe eines Userkeys lässt sich der User mit einem Hub auf dem Server verknüpfen. In der App muss diese dann nur eingegeben werden.



A.4 Hinzufügen von Zonen und den damit verbundenen Sensoren zum Hub

Eine Anleitung hilft dem Nutzer dabei eine Zone hinzuzufügen. Dabei muss man auch den Sensor mit dem Hub verbinden, sobald dieser gefunden wurde, ist der Einrichtungsprozess beendet.



A.8 Anzeige, ob die Batterie der Sensoren leer ist

Falls der Sensor genügend Batterie hat, wird ein volles Batterie-Symbol angezeigt. Bei 20 % Kapazität wird eine Warnmeldung und ein rotes Batterie-Symbol abgebildet.

A.12 Ändern des Namens einer Gießzone

Über das Stift-Icon kann der Nutzer eine Zone umbenennen, dabei darf der Name maximal 12 Zeichen lang sein.