



DRONEDROP

Short-range Lieferung
– per Drohne –

Projektkurs 2024
Lennox, Henk, Linus

ANDROID APP

- Design & Mockup
- Frontend
- Backend

Sichtbarkeit

Öffentlich

Privat

Kategorie

Analyse

Big Data

Datenbanken

Maschinelles Lernen



ChatGPT

Gesundheitswesen

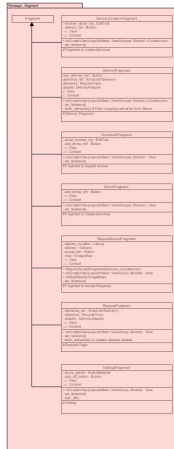
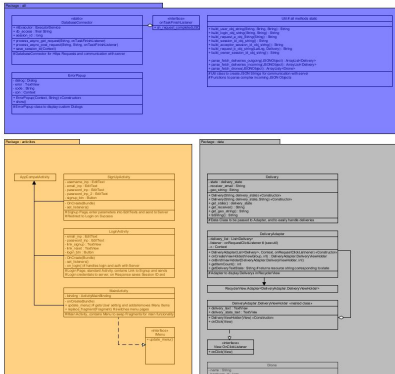
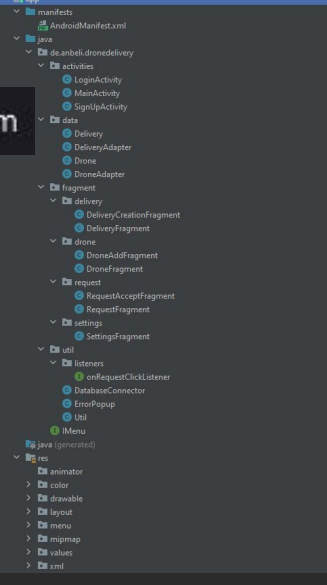
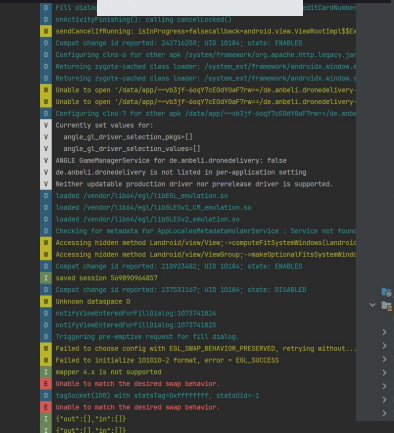
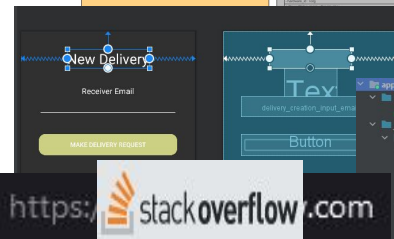
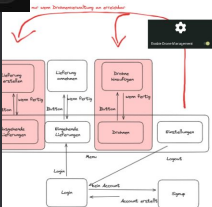
Wissenschaft & Forschung

Google Enterprise APIs


Klima

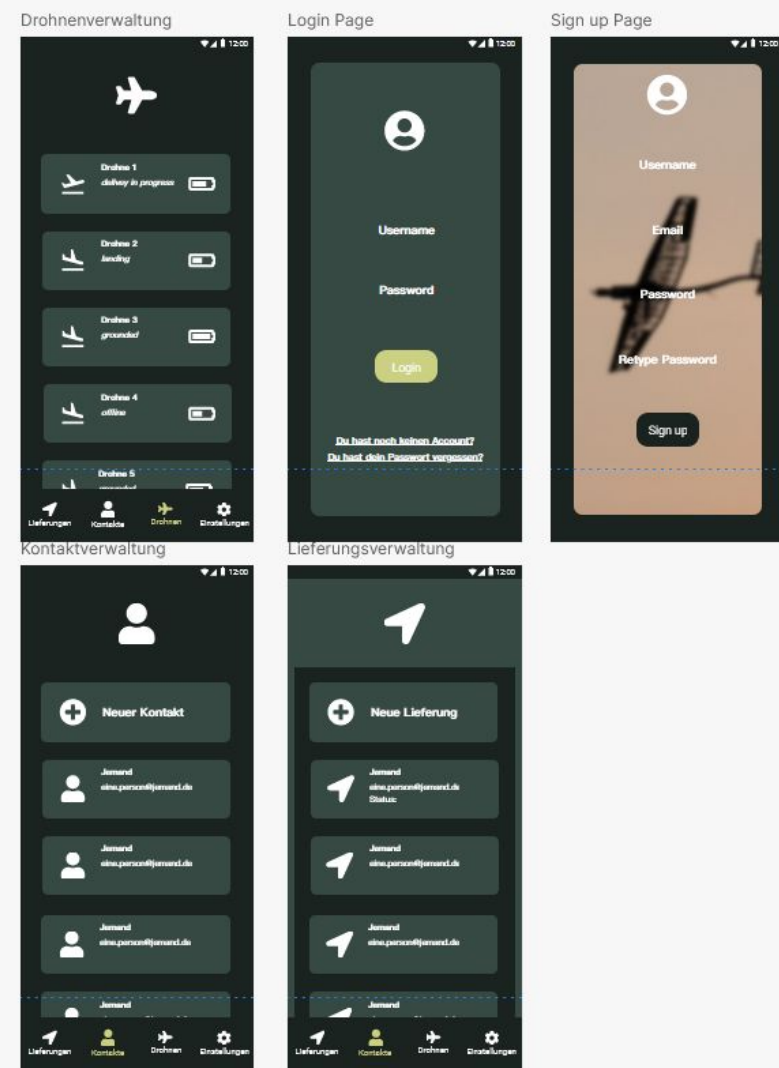
Blog & CMS

Sicherheit



DESIGN & MOCKUP

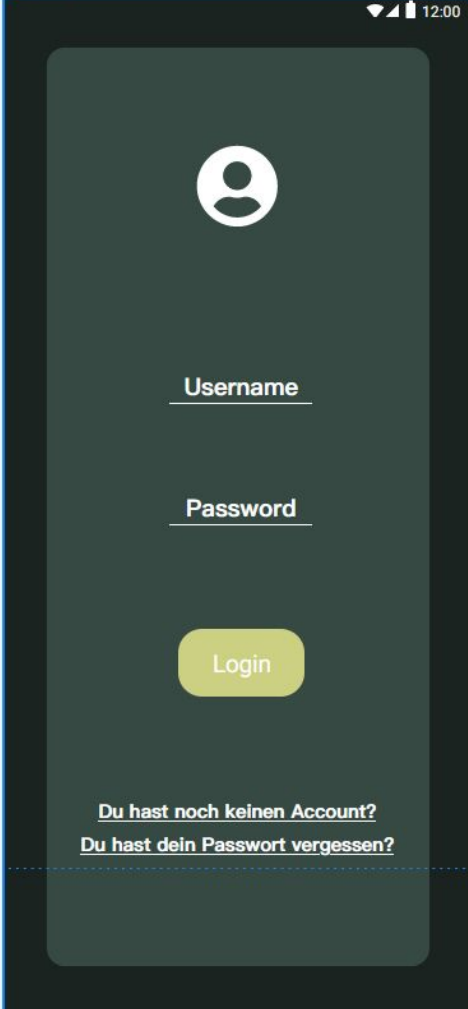
1.  MOCKITT Projekt, erste Vision
2. Anforderungen in Screens:
 - a. Liefern
 - b. Empfangen
 - c. Drohen verwalten
 - d. Einstellungen (hier noch unkonkret)



AUßERHALB MENÜ

- Signup & Login als eigene Screens
- außerhalb restlicher Menüführung

Login Page

A mobile app login screen with a dark teal background. At the top is a white user icon. Below it are input fields for 'Username' and 'Password'. A yellow 'Login' button is centered below the fields. At the bottom, there are two links: 'Du hast noch keinen Account?' and 'Du hast dein Passwort vergessen?'. The status bar at the top shows signal, battery, and the time 12:00.

Username

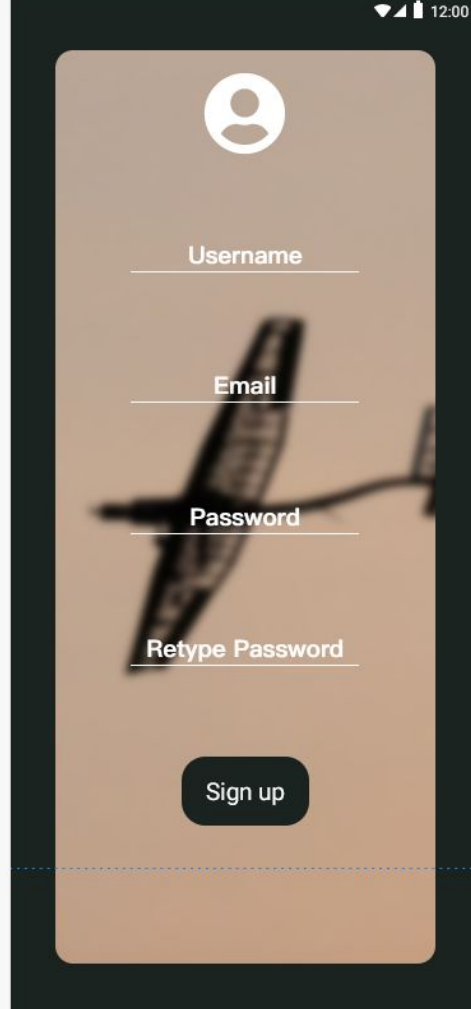
Password

Login

[Du hast noch keinen Account?](#)

[Du hast dein Passwort vergessen?](#)

Sign up Page

A mobile app sign-up screen with a light brown background. At the top is a white user icon. Below it are input fields for 'Username', 'Email', 'Password', and 'Retype Password'. A dark 'Sign up' button is at the bottom. The background features a faint airplane silhouette. The status bar at the top shows signal, battery, and the time 12:00.

Username

Email

Password

Retype Password

Sign up

EIGENTLICHE APP

- Nach Login Navigation über Navigationbar
- Vier Tabs -> NavBar optimal
- simple Screens mit Liste mit Elementen
 - Lieferungen
 - Kontakte (später Anfragen)
 - Drohnen
 - Einstellungen



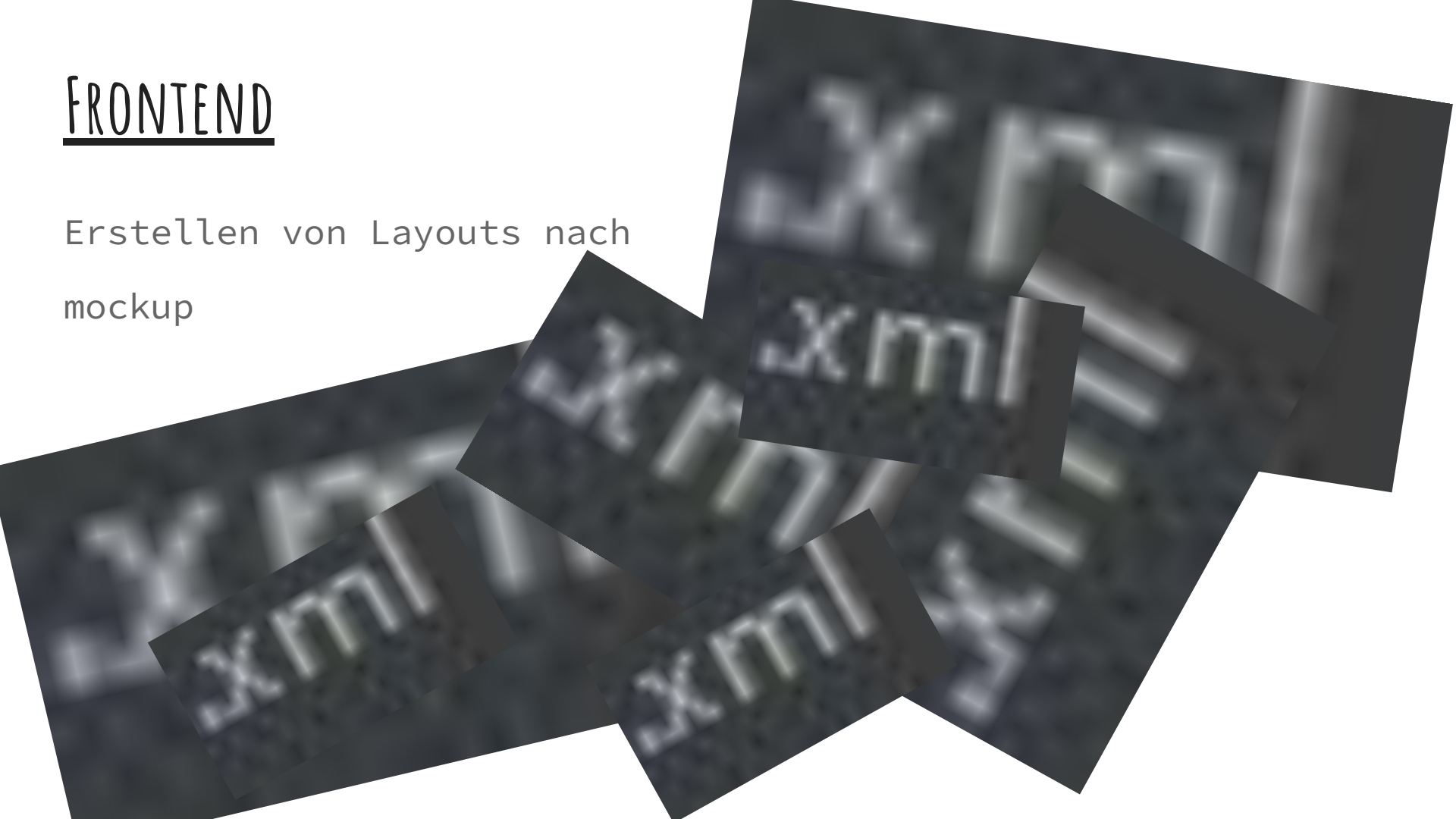
"BRANDING" ;)

- Projektname ursprünglich Dronedeliverys -> zu lang
- Wechsel zu DroneDrop
- schnelles Icon Design



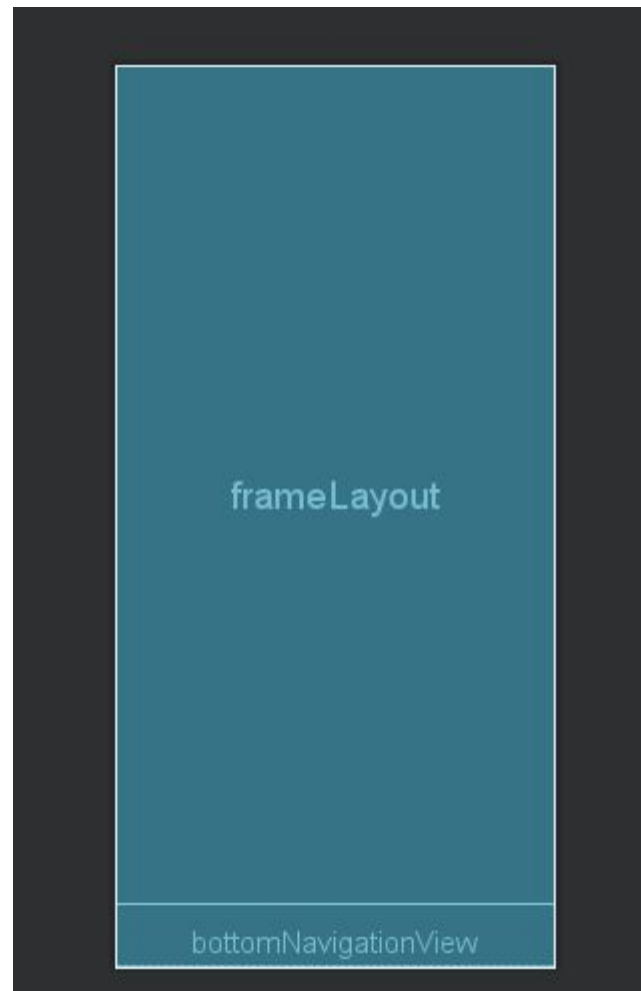
FRONTEND

Erstellen von Layouts nach
mockup



LAYOUTS

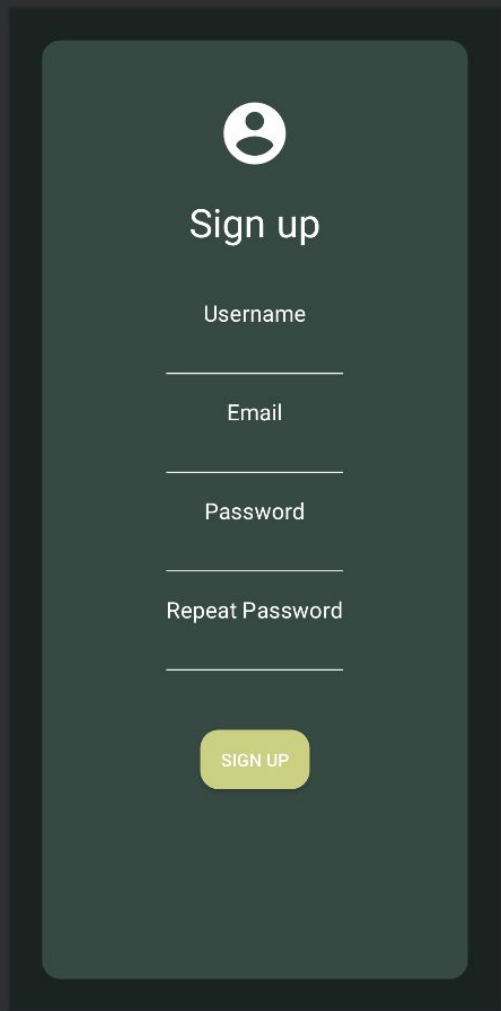
Main Activity mit Menü,
darüber wechselt eigentlicher Inhalt



LOGIN & SIGN UP

Link verweist auf
Sign-up

nach erfolgreichem
Signup zum Login



A dark-themed sign-up form with a white user icon at the top. The title "Sign up" is centered. Below it are four input fields: "Username", "Email", "Password", and "Repeat Password". Each field has a horizontal line for text entry. At the bottom is a yellow "SIGN UP" button.

Sign up

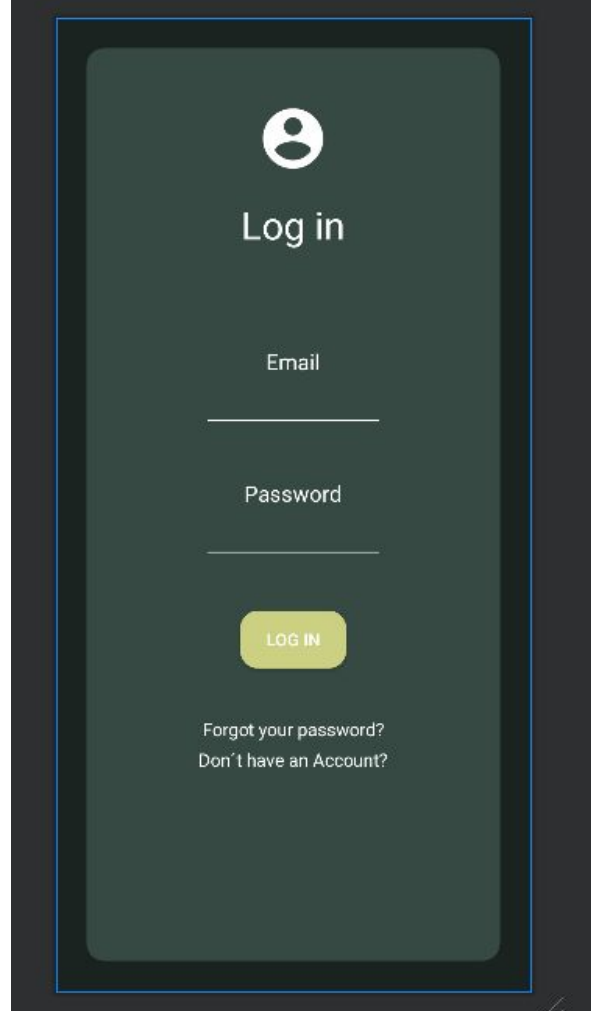
Username

Email

Password

Repeat Password

SIGN UP



A dark-themed log-in form with a white user icon at the top. The title "Log in" is centered. Below it are three input fields: "Email", "Password", and a "LOG IN" button. Each field has a horizontal line for text entry. Below the button are two links: "Forgot your password?" and "Don't have an Account?". The entire form is outlined with a blue border.

Log in

Email

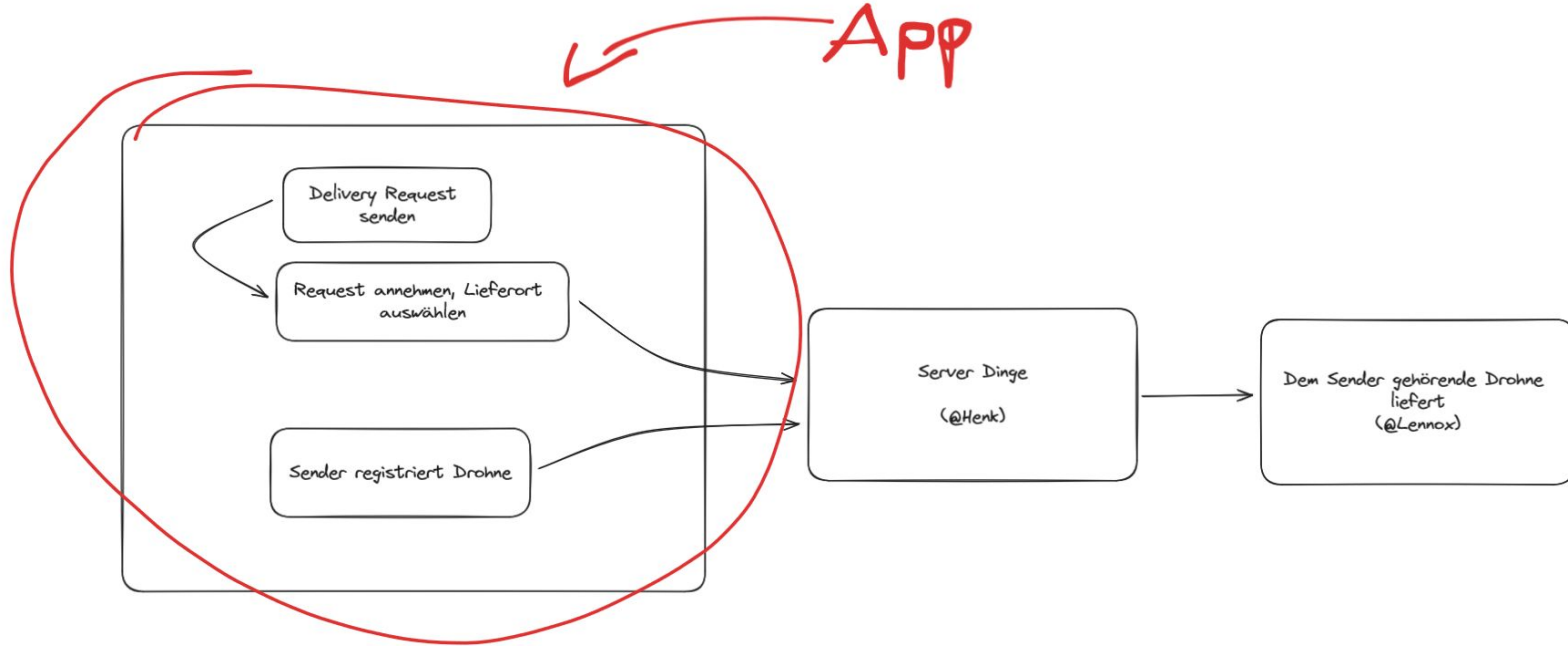
Password

LOG IN

Forgot your password?
Don't have an Account?

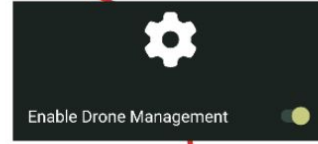
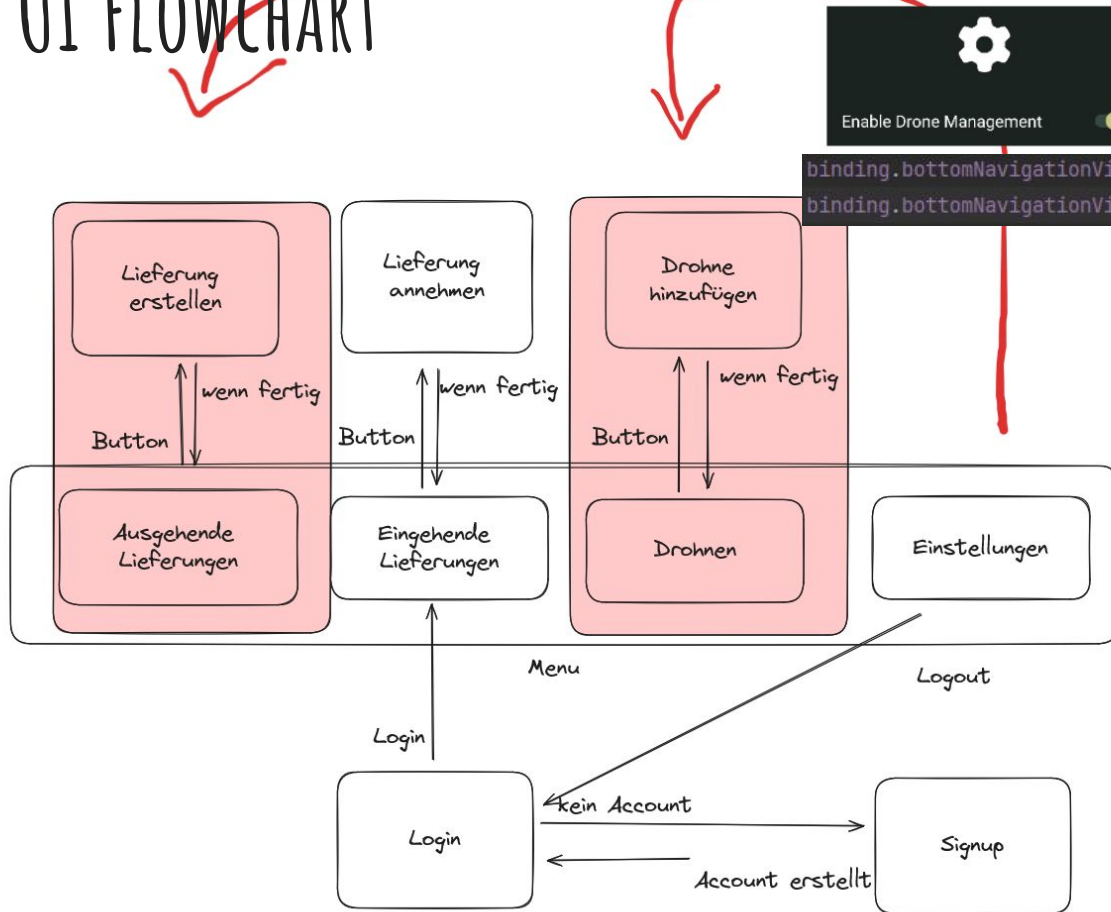
FRAGMENTS

Anforderungen



UI FLOWCHART

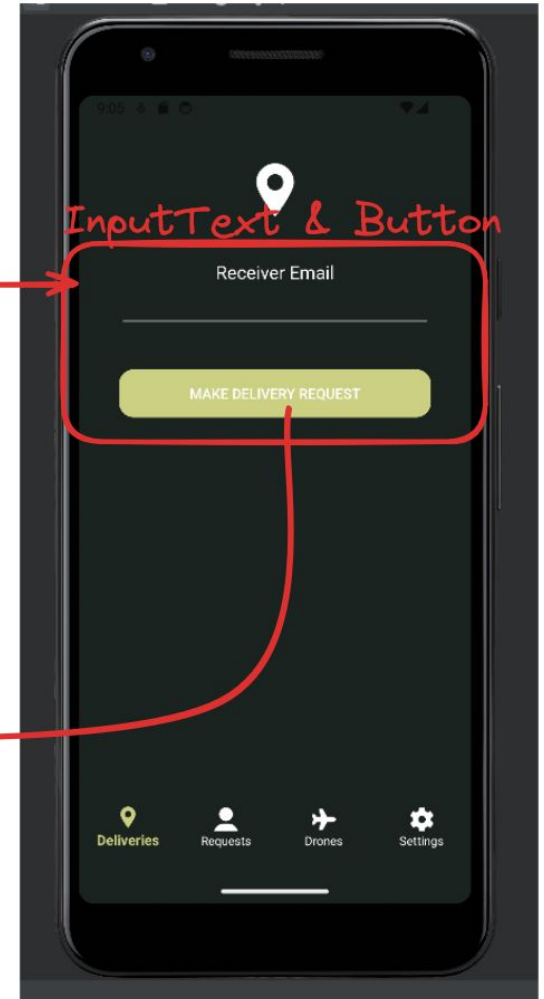
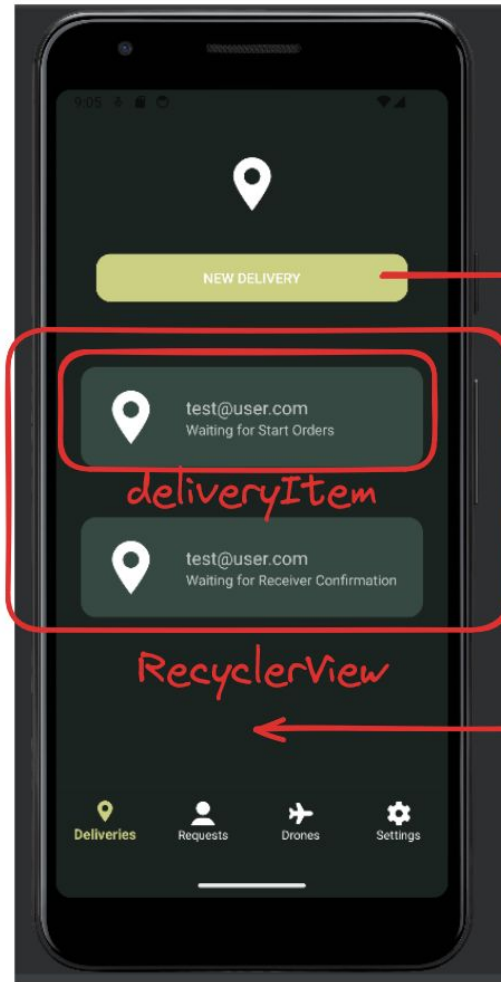
nur wenn Drohnenverwaltung an erreichbar



```
public void update_menu() {
```

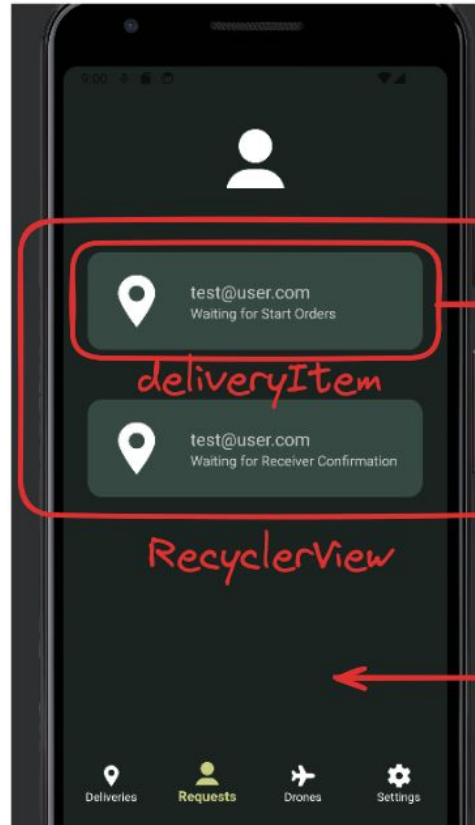
```
binding.bottomNavigationView.getMenu().getItem(index: 0).setVisible(drone_management);  
binding.bottomNavigationView.getMenu().getItem(index: 2).setVisible(drone_management);
```

LIEFERUNGEN



LIEFERANFRAGEN

Lieferung wird bei anklicken übergeben

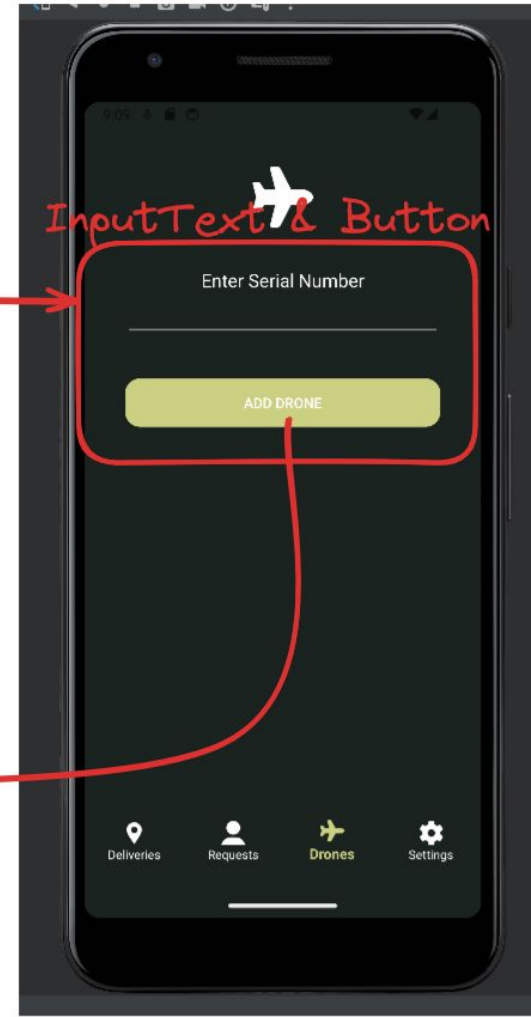
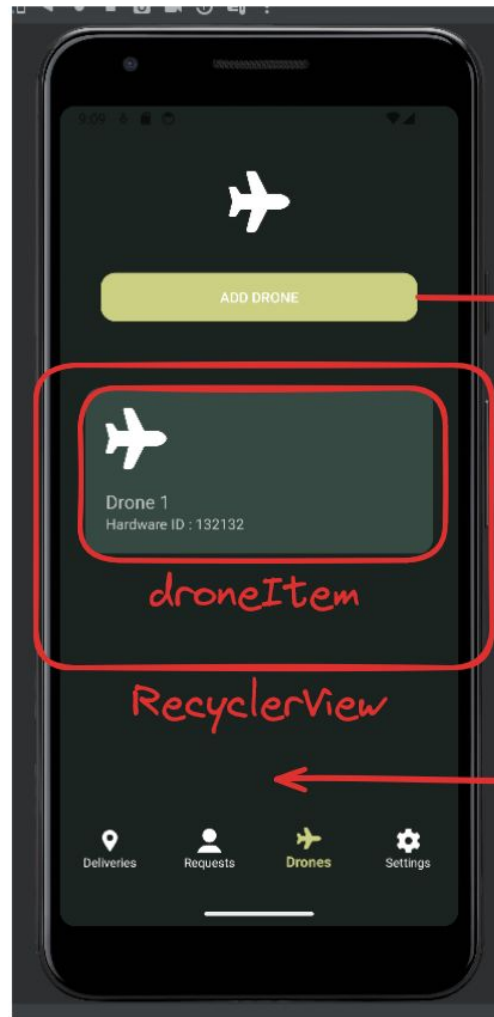


```
//Save the delivery that is being accepted for further processing by passing to constructor  
1 usage  1 an3li  
public RequestAcceptFragment(Delivery delivery) { this.delivery = delivery; }
```

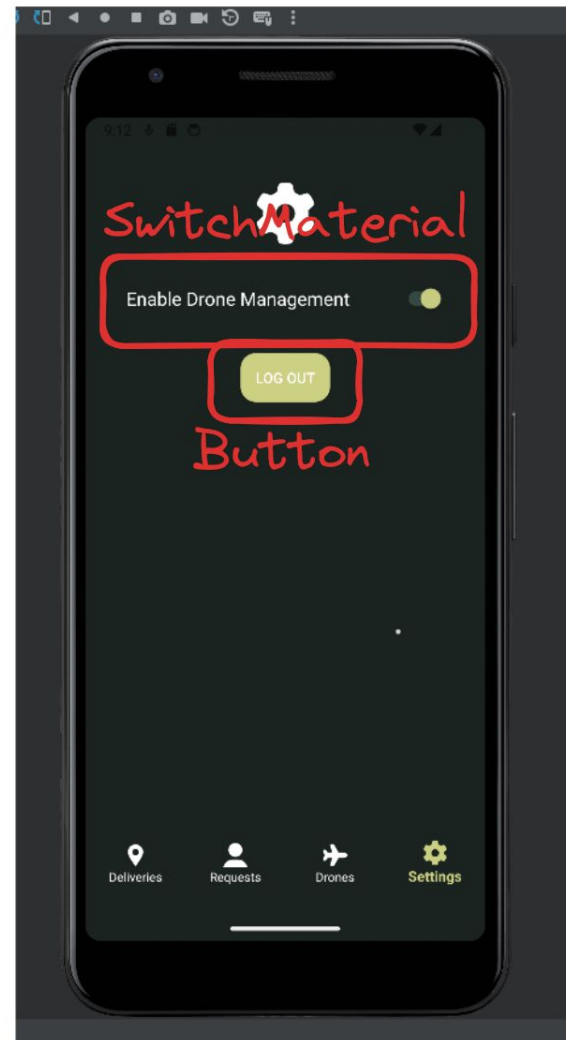
DRONES

aktuelle nur eine
Drohne vorgesehen;

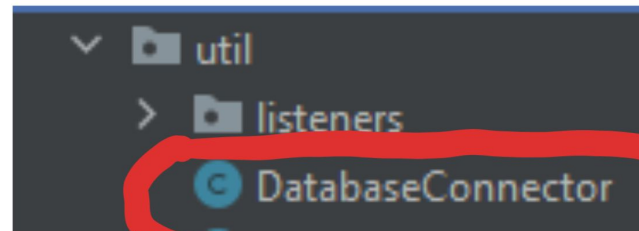
App scalable



SETTINGS



BACKEND DATABASECONNECTOR



THIS GUY

- Session ID handling

```
public static void save_session_id(Context c) {
```

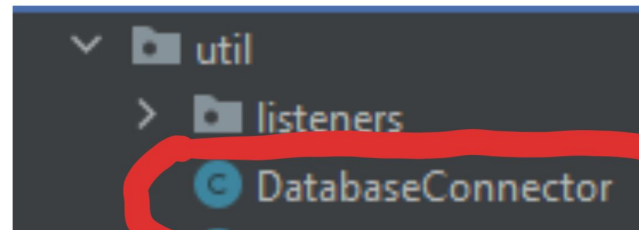
- Requests to Server

```
process_async_post_request(String url_add, String data, onTaskFinishListener listener)  
& get_http_url_connection(String url_add)
```

- Pass back response

```
public interface onTaskFinishListener {  
    1 usage  an33li  
    void on_request_completed(JSONObject res) throws JSONException;  
}
```


BACKEND DATABASECONNECTOR



THIS GUY

```
static ExecutorService mExecutor = Executors.newSingleThreadExecutor();
```

```
//Address of Server
```

```
2 usages
```

```
public static final String db_access = "https://vtol.weyllyn.net/api/";
```

```
14 usages
```

```
public static long session_id = -1;
```

BACKEND DATABASECONNECTOR

Session ID handling

```
public static long session_id = -1;
```

```
public static void save_session_id(Context c) {
```

```
    SharedPreferences.Editor e = c.getSharedPreferences( name: "save_data", MODE_PRIVATE).edit();  
    e.putLong("session_id", DatabaseConnector.session_id);  
    e.apply();  
    System.out.println("put sessionID " + DatabaseConnector.session_id);
```

Außerdem

```
[ private void sign_off() { (in SettingsActivity) ]
```

BACKEND DATABASECONNECTOR



Requests an Server

```
public static void process_async_post_request(String url_add, String data, onTaskFinishListener listener)
```

```
URLConnection con = get_http_url_connection(url_add);  
con.setRequestMethod("POST");  
con.setRequestProperty("Content-Type", "application/json");  
con.setRequestProperty("Accept", "application/json");
```

Verbindung
aufbauen

```
os.write(data.getBytes());  
os.flush();
```

Daten senden

```
while ((inputLine = in.readLine()) != null) {  
    response.append(inputLine);  
}
```

Daten empfangen

```
listener.on_request_completed(new JSONObject(result));
```

Daten über Listener
zurück

BACKEND DATABASECONNECTOR

Requests an Server

Aufbau der zu sendenden Daten

```
1 usage  an3b3li
public static String build_user_obj_string(String name, String mail, String password) {...}
```

```
1 usage  an3b3li
public static String build_login_obj_string(String mail, String password) {...}
```

```
1 usage  an3b3li
public static String build_request_a_obj_string(String receiverEmail) {...}
```

```
4 usages  an3b3li
public static String build_session_id_obj_string() {...}
```

```
1 usage  an3b3li
public static String build_acceptor_session_id_obj_string() {...}
```

```
1 usage  an3b3li
public static String build_request_b_obj_string(LatLng location, Delivery delivery) {...}
```

```
1 usage  an3b3li
public static String build_owner_session_id_obj_string() {...}
```

```
1 usage  an3b3li
public static String build_drone_id_obj_string(int hardwareID) {...}
```

```
1 usage  an3b3li
public static ArrayList<Delivery> parse_fetch_deliveries_outgoing(JSONObject toParse) {...}
```

```
1 usage  an3b3li
public static ArrayList<Delivery> parse_fetch_deliveries_incoming(JSONObject toParse) {...}
```

```
1 usage  an3b3li
public static ArrayList<Drone> parse_fetch_drones(JSONObject toParse) {...}
```

```
1 usage  an3b3li
public static String build_request_b_obj_string(LatLng location, Delivery delivery) {
    String jsonString = null;
    String location_str = location.latitude + "," + location.longitude;

    try {
        jsonString = new JSONObject()
            .put( name: "sessionID", DatabaseConnector.session_id)
            .put( name: "receiver", delivery.get_receiver())
            .put( name: "geoString", location_str)
            .toString();
    } catch (JSONException e) {
    }

    return jsonString;
}
```

BACKEND DATABASECONNECTOR

alles natürlich...

```
static ExecutorService mExecutor  
  
Runnable backgroundRunnable = () -> {  
  
    mExecutor.execute(backgroundRunnable);  
}
```

asynchron :)

GESAMTE APP : UTIL

Package : util

«static»
DatabaseConnector

+ mExecutor : ExecutorService
+ db_access : final String
+ session_id : long

+ process_async_get_request(String, onTaskFinishListener)
+ process_async_post_request(String, String, onTaskFinishListener)
+ save_session_id(Context)

DatabaseConnector for Https Requests and communication with server

«interface»
onTaskFinishListener

+ on_request_completed(JSC

ErrorPopup

- dialog : Dialog
- error : TextView
- code : String
- con : Context

+ ErrorPopup(Context, String) «Constructor»
+ show()

ErrorPopup class to display custom Dialogs

Util # all methods static

+ build_user_obj_string(String, String, String) : String
+ build_login_obj_string(String, String) : String
+ build_request_a_obj_string(String) : String
+ build_session_id_obj_string() : String
+ build_acceptor_session_id_obj_string() : String
+ build_request_b_obj_string(LatLng, Delivery) : String
+ build_owner_session_id_obj_string() : String

+ parse_fetch_deliveries_outgoing(JSONObject) : ArrayList<Delivery>
+ parse_fetch_deliveries_incoming(JSONObject) : ArrayList<Delivery>
+ parse_fetch_drones(JSONObject) : ArrayList<Drone>

Util class to create JSON Strings for communication with server
Functions to parse complex incoming JSON Objects

GESAMTE APP : FRAGMENTS

DeliveryCreationFragment
- receiver_email_inp : EditText - delivery_btn : Button - v : View - c : Context
+ onCreateView(LayoutInflater, ViewGroup, Bundle) «Constructor» - set_listeners()
Fragment to create deliveries

DeliveryFragment
new_delivery_btn : Button deliveries_list : ArrayList<Delivery> deliveries : RecyclerView adapter: DeliveryAdapter v : View c : Context
+ onCreateView(LayoutInflater, ViewGroup, Bundle) «Constructor» - set_listeners() - fetch_deliveries() # Fetch outgoing deliveries from Server
Delivery Fragment

DroneAddFragment
- serial_number_inp : EditText - add_drone_btn : Button - v : View - c : Context
+ onCreateView(LayoutInflater, ViewGroup, Bundle) : View - set_listeners()
Fragment to register drones

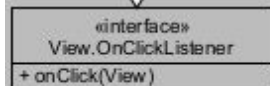
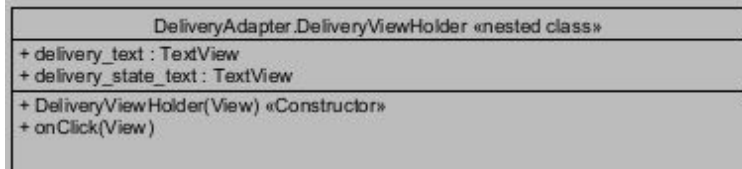
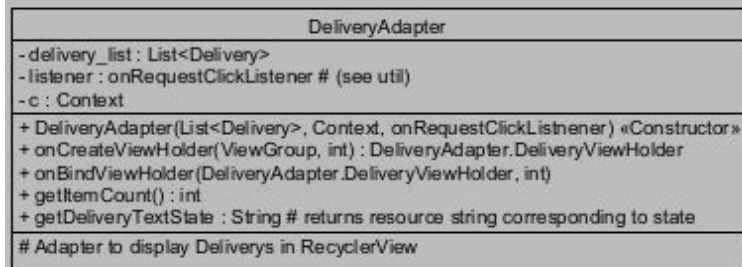
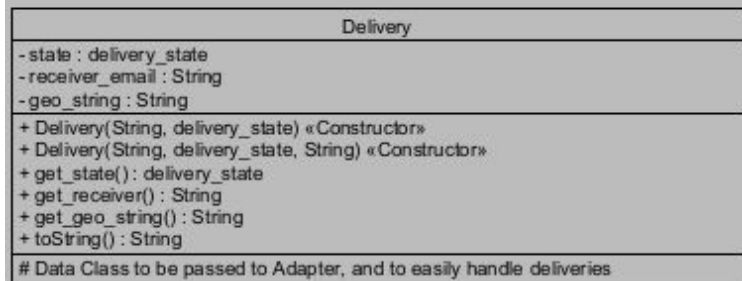
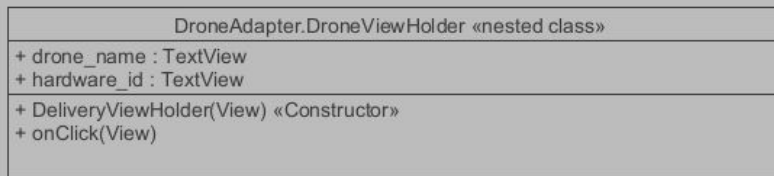
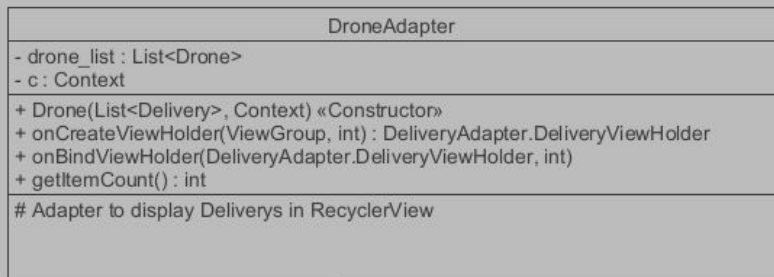
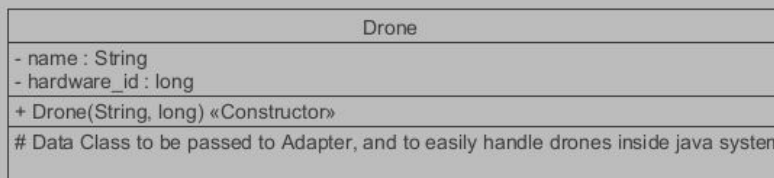
DroneFragment
- add_drone_btn : Button - v : View - c : Context
+ onCreateView(LayoutInflater, ViewGroup, Bundle) : View - set_listeners()
Fragment to create deliveries

RequestAcceptFragment
- delivery_location : LatLng - delivery : Delivery - accept_btn : Button - map : GoogleMap - v : View - c : Context
+ RequestAcceptFragment(Delivery) «Constructor» + onCreateView(LayoutInflater, ViewGroup, Bundle) : View + onMapReady(GoogleMap) - set_listeners()
Fragment to accept Requests

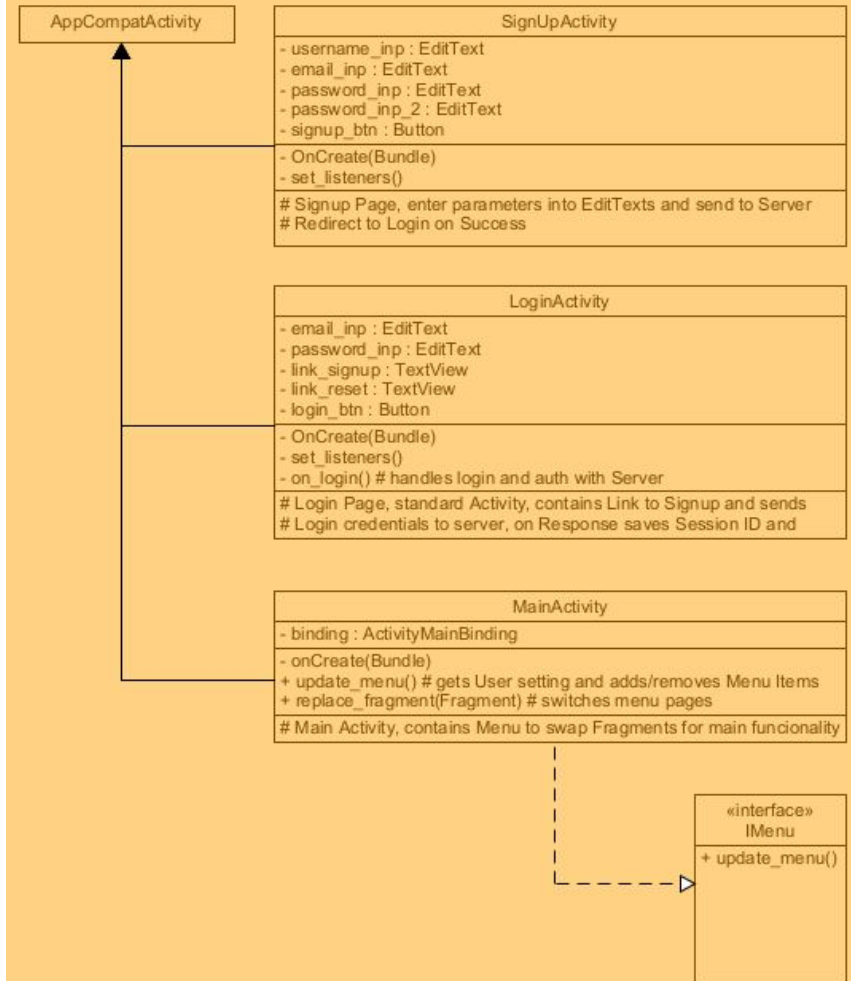
RequestFragment
- deliveries_list : ArrayList<Delivery> - deliveries : RecyclerView - adapter : DeliveryAdapter - v : View - c : Context
+ onCreateView(LayoutInflater, ViewGroup, Bundle) : View - set_listeners() - fetch_deliveries() # updates Adapter dataset
Request Page

SettingsFragment
- drone_switch : SwitchMaterial - sign_off_button : Button - v : View - c : Context
+ onCreateView(LayoutInflater, ViewGroup, Bundle) : View - set_listeners() - sign_off()
Setting

GESAMTE APP : DATA



GESAMTE APP : ACTIVITIES



BITTE TESTEN :)



VTOL UMSETZUNG

- Welche Hardware kommt in Frage?
- 3d Design in Fusion 360
- Ganz viel craften
- Flight Controller
- Stabilisierung und Mixer testen
- Flight Computer

WELCHE HARDWARE?

Anforderungen:

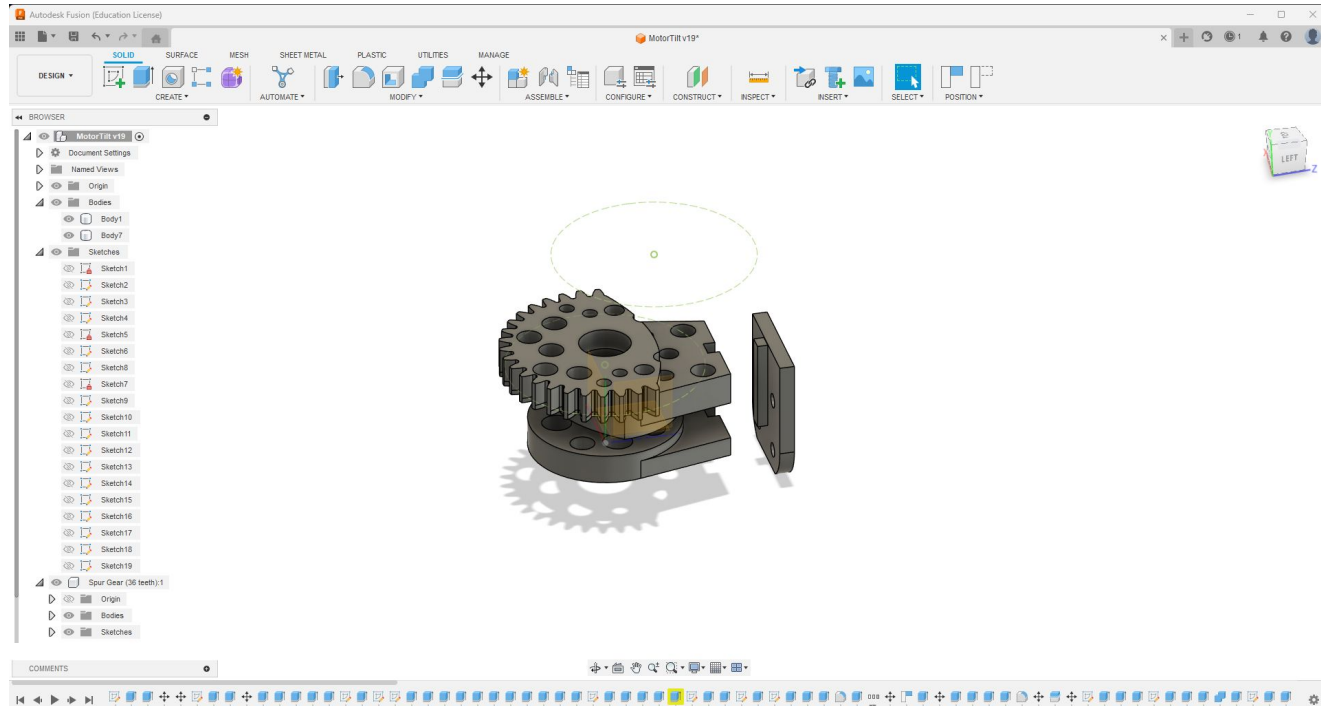
1. Eigenständiges Starten und Landen
2. Reichweite von 10 km
3. Nutzlast von 100g
4. Unter 200 Euro



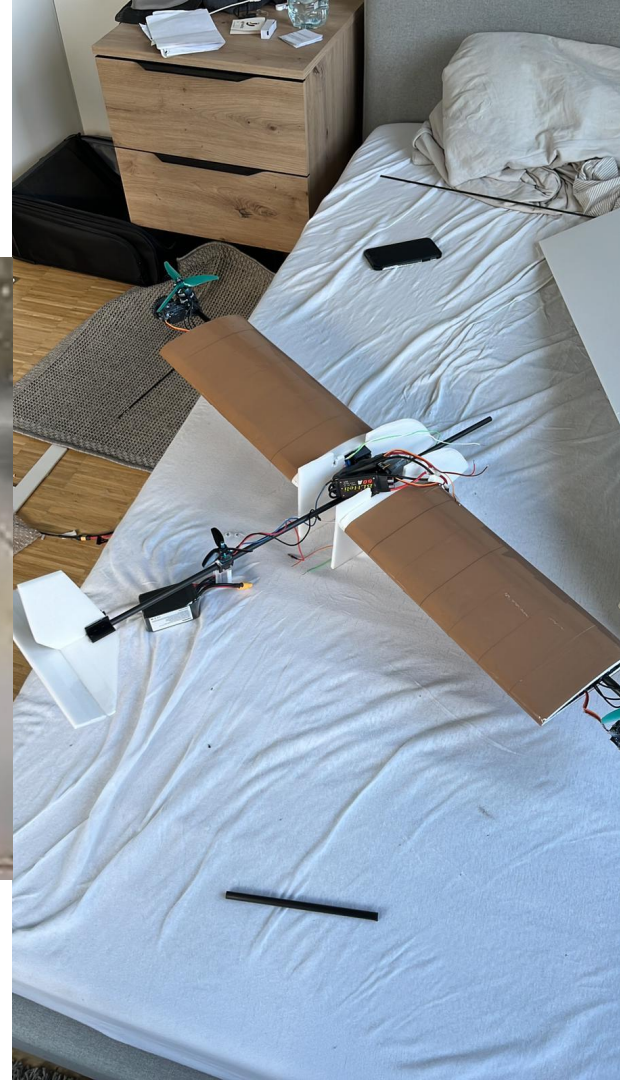
IDEE

- Unnötiges Gewicht minimieren
 - Zwei Motoren
 - Drei Servos
 - Keine Querruder

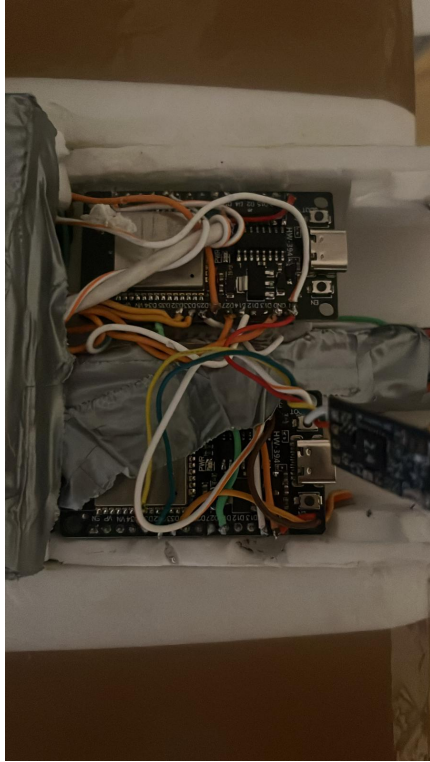
FUSION 360



CRAFTEN



VERBAUTE SENSOREN



F-CON

- Library Madflight (<https://github.com/qqqlab/madflight>)
- Firmware auf Hardware anpassen
- PID Loop
- Mixer auf VTOL anpassen

MADFLIGHT.H

- **Unterstützung für verschiedene Mikrocontroller:** Entwickelt für ESP32, ESP32-S3, RP2040 und STM32, ermöglicht die Bibliothek den Bau von Flugsteuerungen auf diesen Plattformen.
- **Kosteneffizient und DIY-freundlich:** Ermöglicht den Bau einer funktionalen Flugsteuerung für unter \$10 mit leicht verfügbaren Entwicklungsboards und Sensor-Breakout-Boards.
- **Einfacher Einstieg und Anpassung:** Kein kompliziertes Setup oder umfangreiches Lesen von Quellcode notwendig, ideal zum Ausprobieren neuer Flugsteuerungskonzepte.

PID LOOP

- Inputs: desired roll/pitch angle, current angles
- Outputs: correction values roll/pitch

```
PJKurs / vtolArduino / fcon / v10IHOPE.ino
↑ Top

Code Blame 907 lines (757 loc) · 42.5 KB Raw Copy Download Edit View

553 void control_Angle(bool zero_integrators) {
    //desired values
    float roll_des = rcin_roll * maxRoll; //Between -maxRoll and +maxRoll
    float pitch_des = rcin_pitch * maxPitch; //Between -maxPitch and +maxPitch
    float yawRate_des = rcin_yaw * maxYawRate; //Between -maxYawRate and +maxYawRate

    //Serial.printf("r_des:%+.2f p_des:%+.2f y_des:%+.2f",roll_des,pitch_des,yawRate_des);

    //state vars
    static float integral_roll, integral_pitch, error_yaw_prev, integral_yaw;

    //Zero the integrators (used to don't let integrator build if throttle is too low, or to re-start the con
    if(zero_integrators) {
        integral_roll = 0;
        integral_pitch = 0;
        integral_yaw = 0;
    }

    //Roll PID
    float error_roll = roll_des - ahrs_roll;
    integral_roll += error_roll * imu.dt;
    integral_roll = constrain(integral_roll, -i_limit, i_limit); //Saturate integrator to prevent unsafe buil
    float derivative_roll = GyroX;
    roll_PID = 0.01 * (Kp_ro_pi_angle*error_roll + Ki_ro_pi_angle*integral_roll - Kd_ro_pi_angle*derivative_r

    //Pitch PID
    float error_pitch = pitch_des - ahrs_pitch;
    integral_pitch += error_pitch * imu.dt;
    integral_pitch = constrain(integral_pitch, -i_limit, i_limit); //Saturate integrator to prevent unsafe bu
    float derivative_pitch = GyroY;
    pitch_PID = 0.01 * (Kp_ro_pi_angle*error_pitch + Ki_ro_pi_angle*integral_pitch - Kd_ro_pi_angle*derivativ

    //Yaw PID, stablize on rate from GyroZ
    float error_yaw = yawRate_des - GyroZ;
    integral_yaw += error_yaw * imu.dt;
    integral_yaw = constrain(integral_yaw, -i_limit, i_limit); //Saturate integrator to prevent unsafe buildu
    float derivative_yaw = (error_yaw - error_yaw_prev) / imu.dt;
    yaw_PID = 0.01 * (Kp_yaw*error_yaw + Ki_yaw*integral_yaw + Kd_yaw*derivative_yaw); //Scaled by .01 to bri

    //Update derivative variables
    error_yaw_prev = error_yaw;
}
```

TRANSITIONFADEVALUE

- float von 0-1
- Langsamer Übergang zwischen Drohnen-und Flugzeug Modus

```
void control_Mixer() {  
    float throttle_Hover;  
    if(rcin_aux==6){  
        if (rcin_thro >= 0.8) {  
            throttle_Hover = addAndGetSmoothedValue(0.35);  
        } else if (rcin_thro >= 0.6) {  
            throttle_Hover = addAndGetSmoothedValue(0.4);  
        } else if (rcin_thro >= 0.4) {  
            throttle_Hover = addAndGetSmoothedValue(0.35);  
        } else if (rcin_thro >= 0.2) {  
            throttle_Hover = addAndGetSmoothedValue(0.31);  
        } else {  
            throttle_Hover=0;  
        }  
        //Serial.println("Binary Switch: 1.0");  
        //Serial.print("fadedValue");  
        //Serial.println(transitionFadeValue);  
        if(transitionFadeValue<1-tohoverFadeSpeed){  
            transitionFadeValue=transitionFadeValue+tohoverFadeSpeed;  
        }  
    }  
    else if(rcin_aux==3){  
        if(transitionFadeValue<0.5-tohoverFadeSpeed){  
            transitionFadeValue+=tohoverFadeSpeed;  
        } else if(transitionFadeValue>0.5+toplaneFadeSpeed){  
            transitionFadeValue-=toplaneFadeSpeed;  
        }  
    }  
    else{  
        //Serial.println("Binary Switch: 0.0");  
        //Serial.print("fadedValue");  
        //Serial.println(transitionFadeValue);  
        if(transitionFadeValue>0+toplaneFadeSpeed){  
            transitionFadeValue=transitionFadeValue-toplaneFadeSpeed;  
        }  
    }  
}
```

MIXER

```
    }  
}  
  
out_command[MOTOR1] = transitionFadeValue*throttle_Hover +(1-transitionFadeValue)*rcin_thro - transitionFadeValue*(roll_PID + pitch_PID) - (1-transitionFadeValue)*(yaw_PID*0.5);  
out_command[MOTOR2] = transitionFadeValue*throttle_Hover +(1-transitionFadeValue)*rcin_thro + transitionFadeValue*(roll_PID + pitch_PID) + (1-transitionFadeValue)*(yaw_PID*0.5);  
  
//mixing Servos  
if(out_armed) {  
    out_command[SERV01] = (throttle_Hover*0.5-pitch_PID*5)*transitionFadeValue;  
}else{  
    out_command[SERV01] = 0;  
}  
  
out_command[SERV02] = (transitionFadeValue*(yaw_PID*0.5+subTrim3)) + (1-transitionFadeValue)*(-roll_PID+subTrim1) ;  
out_command[SERV03] = (transitionFadeValue*(yaw_PID*0.5+subTrim4)) + (1-transitionFadeValue)*(-roll_PID+subTrim2);  
out_command[SERV04] = 4*pitch_PID + subTrim5;  
}
```

F-COM

- WiFi Verbindung aufbauen
- HTTPS request für Koordinaten
- Navigation

VOID SETUP()

- wird beim startup ausgeführt
- initialisiert sensoren und baut WiFi Verbindung auf

```
void setup() {  
  Serial.begin(9600);  
  if(setUPServer()){  
    Serial.println("SetupServer success");  
    delay(100);  
    Serial.println("FCOMV1: wating for timer");  
    delay(10000);  
    Serial.println("FCOMV1: setupcompass");  
    setUPCompass();  
    Serial.println("FCOMV1: setupPPM");  
    setUPPPM();  
    Serial.println("FCOMV1: setupDistanceSensor");  
    setUPSonic();  
    Serial.println("FCOMV1: setupGPS");  
    setUPGPS();  
    Serial.println("FCOMV1: FoundSats");  
    setUPSERVO();  
    delay(1000);  
    servoDROP(false);  
  }  
}
```


VOID LOOP()

- Wird wiederholt ausgeführt
 - Abhängig von `int FlightState`
 - Zuständig für Webserver
 - Zuständig für Post Request an Server
 - Zuständig für Navigation

<https://github.com/Koefte/PJKurs/blob/main/vtolArduino/fcom/BaseVersion1.ino>

APP-BACKEND

- Was muss der Server/Datenbank können?
- Welche Tools für die Implementation?
- Struktur der Datenbank
- Struktur des Servers
- Verbesserungen/Optimierung

ANFORDERUNGEN AN DAS BACKEND

- Verarbeitung von User-Accounts
 - Sign In (Erstellung von Accounts)
 - Log In (Zuweisung von SessionIDs)
 - Log Out (Löschen der SessionIDS)
 - Abrufen aller User
- Verarbeitung von Drohnen
 - Drohnen einem User zuweisen
 - Drohnen des Users abrufen

- Verarbeitung von Requests

- Ein User stellt Requests an einen anderen
- Ein User empfängt Requests und akzeptiert sie
- Ein User kann seine eingehenden und erstellten Requests abrufen

- Verarbeitung von Deliveries

- Abrufung und Stellung wie bei Requests
- Der Empfänger sendet Koordinaten
- Koordinaten werden von der Drone abgerufen

WELCHE TOOLS/FRAMEWORKS SIND SINNVOLL?



- ExpressJs mit Javascript
 - einfach anzuwenden
 - Verarbeitung von Requestdaten, Statuscodes und Antworten
 - einfach zu hosten und Endpunkte zu definieren
 - persönlich am meisten Erfahrung mit Javascript
 - einfachere, leichtere Implementation
- Andere Tools sind weniger abstrahiert
 - kompliziert, unbekannte Programmiersprachen
 - viel Freiheit => viele Fehlerquellen
- JSON Dateien als Backend
 - Einzelne Tabellen in jeweiligen
 - JSON Files

```
app.get('/api/users', (req, res) => { // User abrufen
  try {
    const data = fs.readFileSync('users.json', 'utf-8');
    // Der Client bekommt ein Array aus allen Usern geschickt, hierbei wird natürlich nicht d
    const jsonData = JSON.parse(data).map(data => ({name: data.name, email: data.email}));
    res.status(200).json(jsonData);
    console.log("Successfully received get request")
  } catch (error) {
    console.error('Error reading users.json:', error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});
```

Backend ER Modell

Serverstruktur

VERBESSERUNGSMÖGLICHKEITEN

- JSON als Backend
 - Scalabilty problematisch
 - Um Elemente einzufügen werden alle Daten in Memory geladen
 - Probleme bei RAM Kapazität bei größeren Anwendungskontexten
- Requeststruktur bei Server
 - Auch Lösbar in dem man weitere Routen/Endpunkte definiert
 - Eventuell so einfacher zu verstehen / erweitern
- Verwaltung mehrerer Drohnen und Deliveries gleichzeitig

DANKE